Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Ananth Mahadevan

Vote Prediction Models for Signed Social Networks

Master's Thesis
Espoo, May 22, 2020

Supervisor: Professor Aristides Gionis
Advisor: Dr. Bruno Ordozgoiti
Examiner: Professor Rohit Babbar

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Ananth Mahadevan		
Title:	Vote Prediction Models for Signed Social Networks		
Date:	May 22, 2020	**Pages:**	81
Major:	Machine Learning, Data Science and Artificial Intelligence	**Code:**	SCI3044
Supervisor:	Professor Aristides Gionis		
Advisor:	Dr. Bruno Ordozgoiti		

Voting is an integral part of the decision-making mechanism in many communities. Voting decides which bills become laws in parliament or users become administrators on Wikipedia. Understanding a voter's behaviour and being able to predict how they will vote can help in selecting better and more successful policies or candidates. As votes tend to be for or against a particular agenda, they can be intuitively represented by positive or negative links respectively in a *signed network*. These signed networks allow us to view voting through the lens of graph theory and network analysis. Predicting a vote translates into predicting the sign of a link in the network. The task of sign prediction in signed networks is well studied and many approaches utilize social theories of balance and status in a network. However, most conventional methods are generic and disregard the iterative nature of voting in communities.

Therefore this thesis proposes two new approaches for solving the task of vote prediction in signed networks. The first is a *graph combination method* that gathers features from multiple auxiliary graphs as well as encoding balance and status theories using triads. Then, it becomes a supervised machine learning problem which can be solved using any general linear model. Second, we propose a novel *iterative method* to learn relationships between users to predict votes. We quantify a network's adherence to status theory using the concept of *agony* and hierarchy in directed networks. Analogously, we use the spectral decomposition of the network to measure its balance. These measures are then used to predict the votes that comply the most with the social theories.

We implement our approaches to predict votes in the elections of administrators in Wikipedia. Our experiments and results on the WIKI-RFA dataset show that the iterative models perform much better than the graph combination model. We analyse the impact of the voting order on the performance of these models. Furthermore, we find that balance theory represents votes in Wikipedia elections better than status theory.

Keywords:	signed networks, balance theory, status theory, Wikipedia, voting models, graphs
Language:	English

Acknowledgements

Firstly, I would like to extend my deepest gratitude to Professor Aristides Gionis for his constant support and guidance. I am also truly thankful to Dr. Bruno Ordozgoiti for the many brainstorming sessions and discussion throughout the thesis. Their wisdom and help are what made this thesis possible.

I would like to thank all the members of the Data Mining Group for the many deep and insightful interactions. The passion, curiosity and motivation that I enjoyed, being part of this group has encouraged me to pursue a career in research.

I extend my gratitude to all the professors at Aalto University for sharing their knowledge and skills.

I thank all my friend from the Macadamia programme for making these two years truly memorable.

Last, but not least, I thank my family for their never-ending love, support and patience.

Espoo, May 22, 2020

Ananth Mahadevan

Abbreviations and Acronyms

RfA	Request for Adminship. The process of promoting a Wikipedia user to an administrator
LSN	Local Signed Network
LR	Logistic Regression
ROC	Receiver Operator Characteristic
PR_{pos}	The Precision-Recall curve for positive label probabilities
PR_{neg}	The Precision-Recall curve for negative label probabilities
AUC-ROC	The area under the ROC curve
AUC-PR_{neg}	The area under the PR_{neg} curve
AUC-PR_{pos}	The area under the PR_{pos} curve

Contents

Abbreviations and Acronyms **4**

1 Introduction **7**
1.1 Thesis Outline 9

2 Background **10**
2.1 Graph Theory 10
2.1.1 Undirected Graphs 10
2.1.2 Directed Graphs 11
2.2 Signed Graphs 12
2.2.1 Balance Theory 14
2.2.2 Status Theory 16
2.3 Link and Sign Prediction 17
2.4 Hierarchy in Directed Networks 18

3 Vote Prediction **21**
3.1 Result versus Vote Prediction 21
3.2 Voting and Signed networks 22
3.3 Linear Combination of Graphs 23
3.4 Local Signed Network 25
3.4.1 Prediction Based on Balance Theory 26
3.4.2 Prediction Based on Status Theory 28
3.4.3 Iterative Prediction Model 33

4 Wikipedia **36**
4.1 Principles of Wikipedia 36
4.2 Formal Organization of Wikipedia 37
4.2.1 Editors 38
4.2.2 Administrators 39
4.2.3 Bureaucrats 40
4.2.4 Arbitration Committee 41

4.3 Request for Adminship 41

5 Implementation **44**
5.1 Datasets 44
5.1.1 Wikipedia RfA Data 45
5.1.2 User Contribution Data 45
5.2 Graph Combination Model 46
5.2.1 Graphs 47
5.2.1.1 Topic Similarly Graph 47
5.2.1.2 Talk and Interaction Graph 47
5.2.1.3 Signed Graph and Triadic Features 48
5.2.2 Data Preparation 49
5.2.3 Supervised Classification 50
5.3 Local Signed Network Models 51
5.3.1 Iterative Balance Model 52
5.3.2 Iterative Status Model 52
5.4 Voting Order Experiments 56
5.5 Evaluation Metrics 57
5.5.1 Receiver Operating Characteristics 57
5.5.2 Precision Recall 58
5.5.3 F1 Score 59

6 Results and Discussion **61**
6.1 Test Dataset Results 61
6.1.1 Graph Combination Model Results 62
6.1.2 Iterative Model Results 65
6.2 Complete WIKI-RFA Results 66
6.2.1 New Voter Analysis 66
6.2.1.1 Iterative Balance Model Results 67
6.2.2 Iterative Status Model Results 68
6.3 Voting Order Results 69
6.3.1 Failed RfA Results 70
6.3.2 Successful RfA Results 71

7 Conclusions and Future Work **73**

Chapter 1

Introduction

In recent years, researchers have become increasingly interested in understanding the behaviour of voters in social networks. Knowledge of the factors that motivate voters, for example, voting for bills in the United States Congress [29] or electing administrators in Wikipedia [10, 28, 35], is of great importance in selecting successful policies or candidates. Voting is a classic problem and has been studied extensively in the fields of game theory and political science [31, 49, 69]. More recently, there is a focus on using information from networks formed from the interaction of voters to model their behaviour. This provides an insight into these interactions and into the influence of certain individuals on voters within a community.

In many communities, decision making is carried out through votes. In these voting sessions, voters indicate if they are for or against an agenda through their vote. These votes can be represented as positive or negative links in a *signed network*. Analysing this network of voting yields various interesting insights. For instance, using methods such as correlation clustering [7, 13, 39] on these signed networks reveals communities that vote similarly and have common ideologies. This provides us with a macroscopic perspective of election dynamics and voting blocks.

On the other hand, predicting the sign of future links in these signed networks gives us a local understanding of the nodes in the network [14, 36, 37]. This task is called *sign prediction* and translates into predicting future votes in the voting networks. The methods to predict the sign of a link use social theories of balance and status. For instance, balance theory states that a friend of an enemy is likely to be an enemy [24]. Therefore, if a user disagrees with a common neighbour who supports an agenda, then they are more likely to oppose that agenda. Similarly, status theory states that people interact on the basis or relative merit [36]. Hence, if you disregard someone who in turn disregards an agenda, then you are more likely to

disregard that agenda. These social theories provide a strong framework, using which we can predict future interactions betweens voters and an agenda within a community. By quantifying agreement or respect in a particular community, we can understand the motivations and factors that affect an individual voter's decision.

The traditional sign prediction approaches are abstract and general, so that they can be applied to signed networks that may not be voting networks [1, 32, 65]. Therefore, they disregard the iterative and chronological nature in which voting usually happens. Furthermore, these methods rely on features obtained from counting triangles or *triads*, to encode the theories of balance and status. Hence, they fail to incorporate larger effects of balance and status in a network. Moreover, in cases where research does focus on sign prediction in voting networks, they heavily rely on non-voting features of the voters and agendas. They utilize these features and build bespoke machine learning models that are task-specific and static [28, 29].

Firstly, in this thesis, we propose a method to extend conventional sign prediction for the task of *independent vote prediction*. For an independent voter, we define the *voting neighbourhood* with respect to a graph and the previous voters in the session. Then, we gather the features from several *auxiliary graphs* that contain non-voting relationships between users. Furthermore, we collect triadic information from the voting network and create a combined feature vector for a voter. Therefore, we formulate a supervised machine learning problem to classify and predict the sign of a vote, using true sign of the votes as targets. This *graph combination model* can be trained with any general linear method using appropriate data processing techniques.

Secondly, we present a novel iterative framework that utilizes theories of balance and status in the *local signed network* (LSN) of a voter to predict the sign of their vote. The framework maintains and constantly updates a *relationship graph*. The edges of this graph capture interaction between voters such as agreement or concurrence. The LSN is defined as the intersection of the relationship graph with the graph of the current voting session. Then, we quantify how much the LSN complies with balance theory or status theory by utilizing the spectral decomposition or the *agony* of the LSN respectively. This allows us to predict the vote as the edge, that when added to the LSN, complies more with either balance or status theory. Therefore, we create two models, namely an *iterative balance model* and an *iterative status model*. These models are iterative as, once the voting is completed in a session, they update their relationship graph with the information from the session. Therefore, these models can be bootstrapped to start with no prior information and improve over time.

Users in Wikipedia undergo a process called a *Request for Adminship*

(RfA), to gain administrative privileges. Candidates are nominated and the RfA is a week long period in which any registered Wikipedia user can show their support for or opposition towards the candidate. After the community finishes its discussions and voting towards the candidate, the result of the RfA is decided by a Bureaucrat (a special class of users). Upon success, the candidate is granted administrative privileges, and upon failure, the candidate can appear for a renomination after a period of time.

We implement the models proposed to predict the votes in Wikipedia RfA elections. The results show that the iterative models far out-perform the graph combination model at predicting votes. We explore the consequences of the voting order on the overall performance of the iterative models. Furthermore, we analyse the features of both models to understand how well the theories of balance and status represent votes in Wikipedia elections and possible scope for future work.

1.1 Thesis Outline

The rest of the thesis is organized in the following manner. Firstly, we discuss the background relating to signed graphs and hierarchy in directed networks in Chapter 2. Next, in Chapter 3 we describe the vote prediction problem and approaches to solving it. Chapter 4 provides a comprehensive view of Wikipedia and the election process for administrators. In Chapter 5 we explain the datasets used, construction of the model and evaluation criteria. After that, we report our findings in Chapter 6 and discuss their implications. Finally, we conclude the thesis and present future work in Chapter 7.

Chapter 2

Background

In this chapter, we provide the fundamentals of the graph theory concepts required to understand the rest of the thesis. In Section 2.1, we cover the basic definitions and terminologies used to describe different types of graphs. Then, we define a signed graph and discuss its unique properties in Section 2.2. We outline the social theories of balance and status in signed networks in Sections 2.2.1 and 2.2.2. Later, in Section 2.3, we provide an overview of the link and sign prediction tasks in signed network and highlights the important approaches from recent works. Lastly, we explain techniques of finding hierarchies in directed networks and the concept of agony in Section 2.4.

2.1 Graph Theory

In this section, we define the various types of graphs and their basic properties. The notation and terminologies used closely follow those used in Diestel [18]. *Graphs* are structures that describe relationships between entities. These entities are called *vertices* and entities related to one another are joined by *edges*. The terms graph, vertices and edges can be used interchangeably with *network*, *nodes* and *links* respectively.

Graphs can be classified broadly into two types based on whether their edges possess a direction. We now go on to define them in detail.

2.1.1 Undirected Graphs

An undirected graph is a pair $G = (V, E)$, where V is the set of vertices and E is the set $E \subseteq \{(u, v) \mid u, v \in V\}$ of unordered pairs of vertices called edges. In this thesis, we will deal only with *simple graphs*, i.e., no self loops, $(u, v) \in V \times V,\ u \neq v$ and there is at most one edge between u and v.

The number of the vertices in a graph is called the *order* of the graph and is denoted by $n = |G|$. In turn, the *size* of a graph is the number of edges denoted by $m = \|G\|$ or $m = |E|$. A vertex u is *adjacent* to v if they are the end points of an edge, $(u, v) \in E$. All the vertices adjacent to a vertex v are called the *neighbourhood* of v and is denoted by $N(v)$. The *degree* of a vertex v is the number of nodes adjacent to that vertex and is denoted by $d(v) = |N(v)|$.

The edges of an undirected graph can have an associated value. This value indicates the distance or similarity between a pair of vertices. These values are called *weights* and the corresponding graph is called a *weighted undirected graph*. Therefore, a weighted graph is defined as a triple $G = (V, E, w)$, where $w : E \to \mathbb{R}^+$ is a function that maps an edge e to a positive real weight $w(e)$. Now, an *unweighted graph* is simply a weighted graph where the function w is defined as: if $e \in E$ then $w(e) = 1$ else $w(e) = 0$. The degree of a vertex v in a weighted graph is the sum of the weights to all the neighbours of v, and is defined as $d(v) = \sum_{u \in N(v)} w((u, v))$. An example of a weighted undirected graph is shown in Figure 2.1.

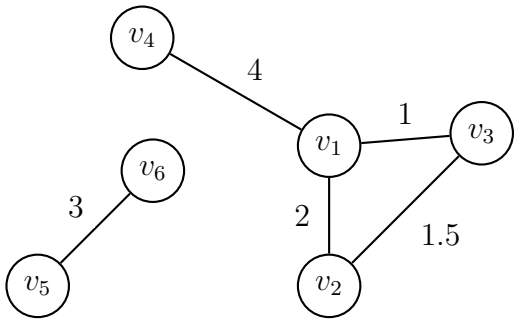


Figure 2.1: An example of a weighted undirected graph

2.1.2 Directed Graphs

The main distinction regarding a *directed graph* (or *digraph*) is that the edges are ordered pairs, i.e., $(u, v) \neq (v, u)$. Therefore, a directed graph has a similar definition: a pair $G = (V, E)$, where V is the set of vertices and E is the set of *ordered* pairs of vertices. Now given an edge $e = (u, v)$ we can define a source function $\text{src} : E \to V$, such that $\text{src}(e) = u$ and a destination function $d : E \to V$, where $\text{dest}(e) = v$. These functions classify the vertices in an edge e as either the source or the destination. In this thesis, we deal only with *simple directed graphs*, i.e., no self-loops, and there can be at most one edge from u to v.

As the edges now have an inherent direction, we can define the *successors* and *predecessors* of a node v. A vertex u is called the *successor* of a node v, if there exists a directed edge from v to u. Therefore, the set of successors for a vertex v is defined as $S(v) = \{u \mid (v, u) \in E\}$. A *predecessor* of a node v is a vertex u such that there exists a directed edge from u to v. The set of predecessors for a vertex v is defined as $P(v) = \{u \mid (u, v) \in E\}$. Now, a vertex u that is either a successor or a predecessor of a vertex v is called a neighbour of the vertex v. Therefore, we define the *neighbourhood* of a vertex v as the set of vertices in the union of successors and predecessor, i.e. $N(v) = S(u) \cup P(v)$. This definition is also compatible with undirected graphs, because if $(u, v) \in E$ then $(v, u) \in E$.

Directed graphs can also have values associated with each directed edge called a *weight*. A *weighted directed graph* can be defined as a triple $G = (V, E, w)$, where the weight function $w : E \to \mathbb{R}^+$ maps each edge e to a weight $w(e)$. Now, the indegree of a vertex v is defined as the sum of the edge weights from the predecessors of v and is denoted as $d_{\text{in}}(v) = \sum_{u \in P(v)} w((u, v))$. Similarly, the outdegree of a vertex v is defined as the sum of the edge weights to the successors of v and is denoted by $d_{\text{out}}(v) = \sum_{u \in S(v)} w((v, u))$. Figure 2.2 shows an example of a weighted directed graph.

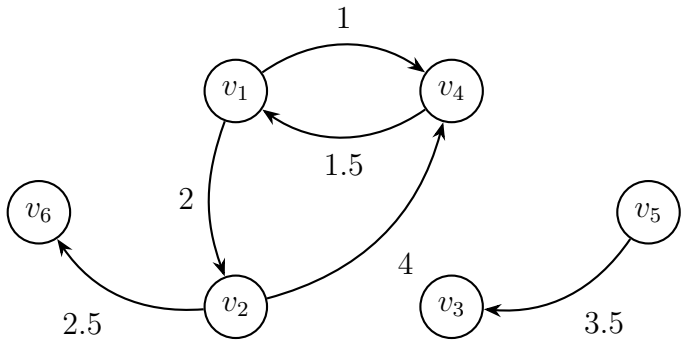


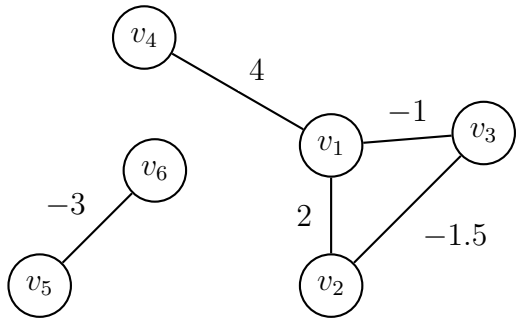
Figure 2.2: An example of a weighted directed graph

2.2 Signed Graphs

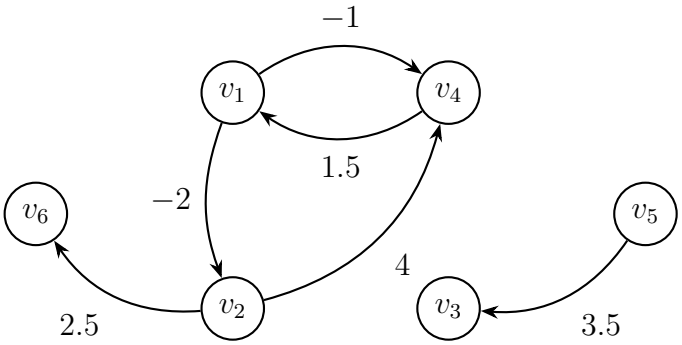
The simple weighted graphs we have defined so far only have non-negative edge weights that can represent similarity or closeness. In the 1950s, social psychologists found it desirable to express liking, disliking or indifference in social interactions. This was formalized by Harary [24] using graphs with weights $(-1, 0, 1)$. These graphs are therefore called *signed graphs*, where a

negative edge weight can denote dissimilarity between a pair of vertices. In this thesis we will use notations and terms similar to Gallier [19], Kunegis et al. [34], Hou [26] and Zaslavsky [68].

A signed graph is a triple $G = (V, E, w)$, where V is the set of vertices, E is the set of pairs of vertices and the weight function $w : E \to \mathbb{R}$. The weight function now takes an edge e and maps it to a signed weight $w(e)$. We can partition the edges into positive and negative edges, $E = E^+ \cup E^-$, where $E^+ = \{e \mid w(e) > 0\}$ and $E^- = \{e \mid w(e) < 0\}$. Similar to Zaslavsky [68], we consider undirected signed graphs and directed signed graphs as distinct and separate entities. We can see some examples of signed graphs in Figure 2.3.



(a) A undirected signed graph



(b) A directed signed graph

Figure 2.3: Examples of Signed Graphs

We now proceed to define a few more terms. The degree of a vertex v is now the sum of the absolute edge weight of its neighbours, called the *signed degree* and is defined as

$$\overline{d}(v) = \sum_{u \in N(v)} |w((u, v))|.$$

This can also be extended to *signed indegree* and *signed outdegree* denoted

by $\overline{d_{\text{in}}}(v)$ and $\overline{d_{\text{out}}}(v)$ and defined as

$$\overline{d_{\text{in}}}(v) = \sum_{u \in P(v)} |w((u,v))|,$$

and

$$\overline{d_{\text{out}}}(v) = \sum_{u \in S(v)} |w((v,u))|.$$

We can create a $n \times n$ square weight matrix W, where each entry w_{ij} is defined as

$$w_{ij} = \begin{cases} w((v_i, v_j)) & \text{if } (v_i, v_j) \in E \\ 0 & \text{if } (v_i, v_j) \notin E. \end{cases}$$

The signed degree matrix $\overline{D}$ is a diagonal matrix consisting of the signed vertex degrees, $\overline{D} = \text{diag}(\overline{d}(v_1), \ldots, d(v_n))$. We can now define the *signed Laplacian*, $\overline{L}$ as

$$\overline{L} = \overline{D} - W$$

The signed Laplacian along with results from spectral analysis of signed graphs [26, 34], will be useful for balance theory of signed graphs.

2.2.1 Balance Theory

In the 1940s, Heider [25] proposed that when there are either *positive relations* (friendship, love, support) and *negative relations* (enmity, hate, oppose) in a group, the group tends towards *balance*. Balance is the concept that all members aim to maintain consistency in the relations they share with other members of the group. Hence, in a group of three members this can be seen as having three positive relations or, one positive and two negative relations. In Figure 2.4, the triads B_1 and B_2 are *balanced* and mirror aphorisms such as "the friend of my friend is also a friend" and "the enemy of my enemy is a friend" respectively. Therefore, triads B_3 and B_4 are called *unbalanced* or *imbalanced* that denotes the cognitive dissonance between the members of the triad. The dissonance can be understood by the counter-intuitive nature of the phrase "the friend of my friend is my enemy" described by triad B_3 for the node v_1.

Harary and Cartwright [11] generalized this notion of balance by using signed graphs. As these relations are typically symmetric, balance is usually defined for undirected signed graphs. This can been seen in Figure 2.4 where solid edges are positive and dashed edges are negative. Davies [15] offers an

alternate definition named *weak balance*,where triads of type B_2 are considered to have lesser predictive utility as relationships of these type are less common in real-life social networks.

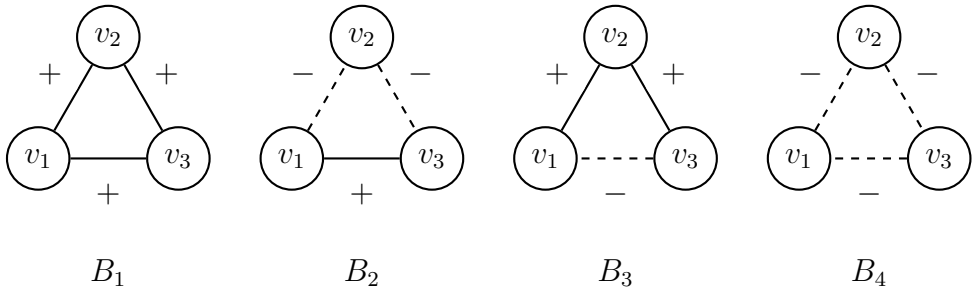


Figure 2.4: Triads in undirected signed graphs. B_1 and B_2 are *balanced* triads as they have even number of negative edges. B_3 and B_4 are *unbalanced* as they have odd number of negative edges.

The concept of balance can be generalized for an undirected signed graph $G = (V, E, w)$. The sign of a cycle C in the signed graph G is defined as the product of the edge weights, $sgn(C) = \prod_{e \in C} w(e)$. A signed network G is then said to be balanced if and only if all the sign of all cycles in network are positive. Therefore, every cycle in G must have an even number of negative edges. This leads to the result from Harary [24] that states that if a graph G is balanced, then there is a partition of the vertices $V = V_1 \cup V_2$ such that edges within the vertices of each set is positive and edges between the sets are negative. This means that once we delete the positive edges in a balanced network, it becomes a bipartite graph. An example of a balanced signed graph is shown in Figure 2.5. Here the partition of the vertex set is $V_1 = \{v_1, v_3, v_4, v_7, v_8\}$ and $V_2 = \{v_2, v_5, v_6, v_9\}$.

The signed Laplacian matrix $\overline{L}$ of a signed graph G is always positive-semidefinite. If the signed graph G is *unbalanced*, then it does not possess a partition that can lead to a bipartite network connected by negative edges. If the smallest eigenvalue of a graph G is denoted by $\lambda_1(G)$, then G is balanced iff $\lambda_1(G) = 0$. Therefore, $\overline{L}$ is positive-definite if and only if G is *unbalanced* [26, 34, 68]. Hou [26] provides bounds on the value of $\lambda_1(G)$ and Li et al. [40] show that $\lambda_1(G)$ can be used as a measure of how far the signed graph G is from being balanced. Therefore, we can use the smallest eigenvalue of the signed Laplacian, $\lambda_1(G)$ as a quantification of the balance of the signed network G.

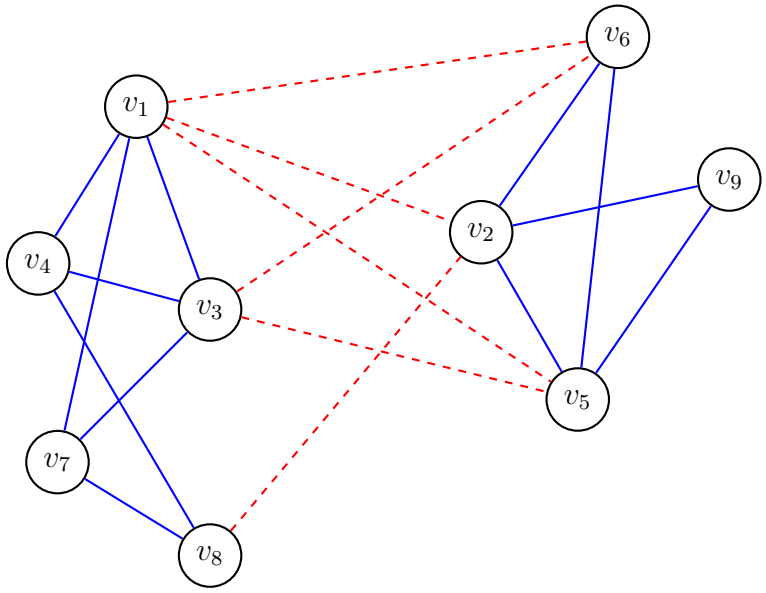


Figure 2.5: A balanced signed graph. Solid blue edges are positive and dashed red edges are negative. Every cycle in this graph contains an even number of negative edges.

2.2.2 Status Theory

Guha et al. [22] mention implicitly that a signed edge from u to v can be interpreted in an asymmetric manner different from "friend" or "enemy". Leskovec et al. [36, 37] introduce the concept of *status* to contextualize directed signed edges. A positive edge $u \xrightarrow{+} v$ indicates that v has a higher status than u and a negative edge $u \xrightarrow{-} v$ means that v has a lower status than u. This concept of relative status can be propagated transitively along multiple steps which might lead to contradictions with balance theory [37].

For instance, given three vertices v_1, v_2 and v_3, the presence of an edge $v_1 \xrightarrow{+} v_2$ indicates that v_1 thinks v_2 has higher status, the edge $v_2 \xrightarrow{+} v_3$ indicates that v_2 thinks v_3 has higher status. Now, we wish to to close this triad with an edge from v_3 to v_1. Status theory would say that through transitivity, v_1 has lower status than v_3, therefore the prediction is $v_3 \xrightarrow{-} v_1$. Whereas, in balance theory we would predict a positive edge $v_3 \xrightarrow{+} v_1$ to make the cycle have even number of negative edges. In Figure 2.6, S_1 is the triad as predicted by balance theory, while triad S_2 is what status theory predicts.

There are also cases when status theory is ambivalent to the edge that closes a triad. Consider the example when we have the edges $v_1 \xrightarrow{+} v_2$ and

$v_2 \xrightarrow{-} v_3$. If we indicate the status of a vertex v using the function $\sigma(v)$, then the edges describe the following : $\sigma(v_2) > s(v_1)$ and $\sigma(v_2) > \sigma(v_3)$. Therefore, we have no knowledge of the relative difference in status between the vertices v_1 and v_3. Hence, both edges $v_3 \xrightarrow{+} v_1$ ($\sigma(v_3) > \sigma(v_1)$) and $v_3 \xrightarrow{-} v_1$ ($\sigma(v_1) > \sigma(v_3)$) are equally valid for status theory. However, balance theory unequivocally predicts $v_3 \xrightarrow{-} v_1$, to keep the resultant triad balanced. This case is shown in Figure 2.6 as triads S_3 and S_4.

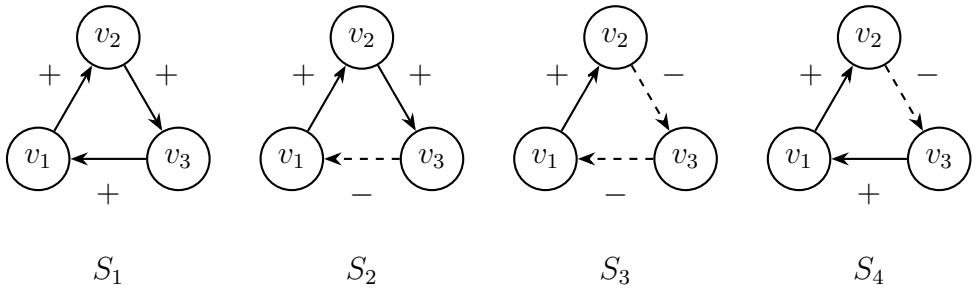


Figure 2.6: Triads in directed signed graphs. Triads S_2, S_3 and S_4 are compliant with status theory. Only triads S_1 and S_3 are compliant with balance theory.

Each positive link inwards ($d^+_{\text{in}}(v)$) and negative link outwards ($d^-_{\text{out}}(v)$) increases status. In turn, each positive link outwards ($d^+_{\text{out}}(v)$) and negative link inwards ($d^-_{\text{in}}(v)$) decreases status. Therefore, $\sigma(v) = d^+_{\text{in}}(v) + d^-_{\text{out}}(v) - d^+_{\text{out}}(v) - d^-_{\text{in}}(v)$ is a heuristic for the status of a node [36]. An interesting fact is that, the edge $u \xrightarrow{-} v$ can be converted into positive edge in the opposite direction $u \xleftarrow{+} v$. This fact reduces the number of unique triads that can be formed using status theory and will be used in edge prediction tasks that will be discussed in the coming chapters.

2.3 Link and Sign Prediction

The *link prediction problem* as defined by Liben-Nowell and Kleinberg [41] is the task of inferring possible future edges between vertices in a social network. To this end, the datasets used were split into training edges and testing edges which had a common core set of vertices. The goal was to use information in the training edges to predict the edges that were likely in the test dataset. They showed that topological features such as number of common neighbours and Katz's centrality index can be used in an unsupervised setting to accurately predict future edges.

Leskovec et al. [36] extended the link prediction problem to signed networks. They also introduced the problem of *edge sign prediction*: predict the sign of a given edge (u, v), using the existing signed network G. A supervised model for the task is proposed that uses network features such as indegree and outdegree of a node along with *triad* features. The edge nodes u and v and a common neighbour w form a triad. The directed edge between (u, w) and (w, u) can be either forward or backwards and each edge is either positive or negative. Therefore, there are 16 possible triad types for a given neighbour w. Hence, for a given edge (u, v), they count the type of triad formed from each common neighbour w and use it as a feature. They analyse these triadic features from the trained model and show how they relate to balance and status theory. Furthermore, for the link prediction problem, the information from the negative edges in the signed network improves the overall accuracy of the model. This seminal paper inspired many more approaches to solving these problems for signed graphs.

Matrix factorization and latent space approaches facilitate link prediction and sign prediction tasks for multiple edges simultaneously in a signed network [1, 21, 27]. Supervised algorithms for both tasks can be improved by utilizing additional node features such as inverse square metric [2] or node rankings [48]. Models using graph motifs as features [32, 42] extend the concept of triadic features for link and sign prediction. Chiang et al. generalize balance theory for longer cycles and incorporate it as features to improve link prediction [14]. Tang et al. [50] discuss the importance of predicting negative links and highlight methods to overcome the inherent imbalance present in signed network datasets. Cesa et al. [12] and Chiang et al. [13] utilize clustering techniques to solve sign prediction and link prediction respectively. Shuang et al. [65] and Karimi et al. [29] create bespoke models incorporating user behaviour and political party affiliation respectively to predict the sign of edges present in the networks.

2.4 Hierarchy in Directed Networks

Hierarchies exist in all social structures. From the explicit levels found in nature, such as the food chain or organizational structures in businesses to more implicit stratification that occurs on social media or online networks. A common method to represent such hierarchies is through a tree. For example, the chain of command in the military or within governments. Trees have well defined levels and a single person at the top. Generalizing this structural concept, we get a *Directed Acyclic Graph* (DAG), which represents a partially ordered set. DAGs have perfect hierarchy, while structures such as cycles

tend to have no hierarchy. Regular directed graphs occur somewhere between these two extremes.

Gupte et al. [23] provide a method to discern the levels of stratification present in a given directed network when no prior information of the hierarchy exists. They define a measure of the hierarchy of a given directed network G as $h(G)$. In turn, they propose a polynomial algorithm to find the largest hierarchy in that network.

They define the concept of *social agony*, which posits that agony is present when a person having a higher rank in network interacts with a person who has a lower rank. Therefore, if we define the rank of a node in graph G as the function $r : V \to \mathbb{N}$, then a directed edge $u \to v$ causes agony when $r(u) \geq r(v)$. The agony of an edge can be quantified as $\max(r(u)-r(v)+1, 0)$. Hence, the agony of the graph G with respect to the rank function r is defined as

$$A(G, r) = \sum_{(u,v)\in E} \max(r(u) - r(v) + 1, 0) \ .$$

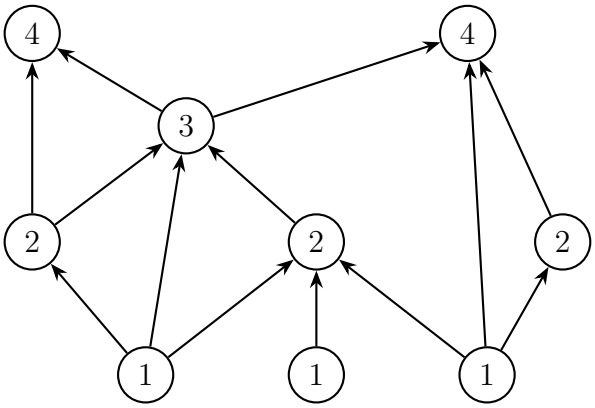
As nodes in a graph tend to minimize the overall agony, the agony of a network G is the smallest possible agony over all possible ranking for r, $A(G) = \min_{r\in \text{Rankings}} A(G, r)$. Therefore, the hierarchy of a given network G, $h(G)$ can now be expressed in terms of the agony of the network

$$h(G) = 1 - \frac{1}{m} A(G),$$

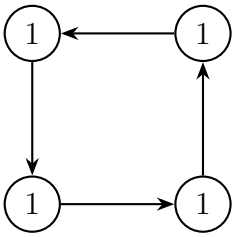
where m is the number of edges. We can see examples of hierarchy in unweighted directed graph in Figure 2.7.

Therefore, finding a ranking of the nodes that minimizes the agony of the network gives us the optimal hierarchy present in that network. Gupte et al. [23] and Tatti [51] present a polynomial algorithm that can solve the dual of the agony minimization problem to obtain the optimal ranking r for unweighted graphs. Tatti [52] provides an alternate approach using a capacitated circulation solver that can handle weighted digraphs as well as additional cardinality constraints. These algorithms allows us to find the levels of hierarchy present in any given directed social network and analyse the interaction between members belonging to different strata in that community.

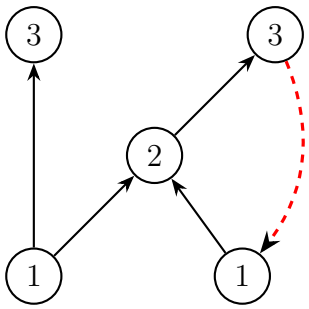
We explain in future chapters how the hierarchy in a social network is intrinsically linked to the theory of status in directed signed networks and how agony can be used to measure the degree of adherence to status theory.



(a) DAG has perfect hierarchy, $h(G) = 1$ and agony of each edge is 0



(b) A cycle has no hierarchy, $h(G) = 0$ and each edge has agony of 1



(c) Graph with some hierarchy $h(G) = 2/5$. Red dashed edge has agony of 3 and solid black edges have 0 agony.

Figure 2.7: Examples of hierarchy in unweighted directed graphs. Numbers inside nodes indicate the rank of vertex.

Chapter 3

Vote Prediction

In this chapter, we cover the main motivation behind predicting the vote of an individual voter and present the methods that can be used to solve this task. First, we discuss the contrast in perspectives that is present when predicting the result compared to predicting a vote in Section 3.1. Next, in Section 3.2, we explain how the problem of vote prediction is intrinsically linked to the tasks of link and sign prediction in a signed network. In Section 3.3, we describe a supervised machine learning framework that uses features from graphs for voting and non-voting data. Lastly, we present our novel approach of constructing a local signed graph of the current voter and using balance and status theory to predict the vote in Section 3.4.

3.1 Result versus Vote Prediction

In this thesis, we are interested in the voting behaviour for a collective action. In such cases, members of a community come together as *voters* to decide on a particular *candidate* item during a *session*. In a parliament the voters are the elected members of the parliament and the candidate is usually a bill or a policy matter. When it is promotion within a political party or an online community such as Wikipedia, the members vote on a candidate who has been nominated for the position. In all these cases, we have two levels of decisions being made. The first is the individual decision that each voter makes with regard to the candidate. The second is the final decision that the group arrives at after everyone has voted. We refer to task of predicting the former as *vote prediction* and the latter as *result prediction*.

Result prediction provides a macro level perspective of the incentives of a community. We can create models based on the characteristics of a candidate to predict the result of a collective action. This leads to an understanding on a

communal level, of what characteristics are preferred and if there are voting blocks formed within the members based on ideological differences. This translates to practical examples such as party level dynamics in a parliament, the topic of a bill or the credentials of a nominee [9, 66, 67].

On the other hand, when we focus on the vote prediction problem, we get a deeper understanding of the dynamics between voters and the candidate. In fields such as game theory, this is well studied using frameworks such as *strategic voting models* and *momentum* [3, 6, 44, 49, 69]. These models are more theoretical and are studied under synthetic conditions. Nevertheless, they still provide a foundation on which we can construct practical models that can utilize additional features. One such popular approach is using textual information from bills, speeches and legislature to predict votes of politicians in parliament [8, 20]. The next important step is to represent the voting data as networks and leveraging network features to understand and predict voter behaviour.

3.2 Voting and Signed networks

Votes by nature express a positive or negative relationship between a *voter* and a *candidate*. Therefore, *signed graphs* provide an intuitive way to structurally represent the voting pattern of members in a community. These signed voting networks can be used to develop models to solve the task of vote prediction and analyse voter behaviour.

Correlation clustering and community detection of signed voting graphs can discover trends and vote blocks in communities [4, 7]. Analysing the networks using social theories of balance and status provides knowledge of voter behaviour and features for prediction methods [16, 39]. The vote prediction task can be broken down into two subtasks which are analogous to link and sign prediction in signed networks.

The first subtask is to predict *who* will vote next given a candidate c and a set of previous votes. This subtask is similar to link prediction in a signed network, where we aim to predict possible future edges of the type (v, c). The complexity of this subtask depends on the format of voting that takes place. For instance, if there is a known voting order, such as roll calls in parliamentary proceedings or explicit timestamps, then the subtask is trivially solved. On the other hand, if the voting occurs simultaneously, the subtask can be reduced to predicting the possible subset of members who will vote in a given session. However, when the voting is iterative and there is no known underlying process of who votes next, then a separate model might be required to infer the voting sequence. This particular case can be

combinatorially hard, as we would need to find the correct ordering of votes in a session.

The second subtask is to predict *how* a voter v will vote for a candidate c given the previous votes in the session. This task translates into predicting the sign of an edge (v, c), in the current session. We call this subtask the *independent vote prediction* problem. It is independent as we are only interested in the decision of the voter v, assuming that we have complete prior knowledge of how the previous votes have been cast. Then, this problem can be framed as a supervised learning task. It uses features of the interaction between the current voter v, previous voters U and the candidate c to predict the sign of the edge (v, c). We can utilize the theories of balance and status in signed networks to create models similar to those by Karimi et al. [29] and Jankowski-Lorek et al. [28] to predict voter behaviour.

3.3 Linear Combination of Graphs

In this section we explain how the approaches outlined in Section 2.3 can be applied to solve the *independent vote prediction* problem. As discussed previously, the *edge sign prediction* task in signed network is analogous to vote prediction. The models proposed by Leskovec et al. [36] can be used to predict the sign of the edge (v, c). However, voting in a community takes place across many sessions in a chronological manner. Therefore, we must partition the training and testing datasets to avoid data leakage. We propose the following framework to gather features using a linear combination of the voting history and several auxiliary graphs.

We denote the directed signed graph for the current voting session as $S = (V_S, E_S, w_S)$. The current voter in consideration is denoted by v and the candidate of the session is c. In this thesis, for all the models we assume that each session has a unique candidate. The votes that have been cast prior to the current voter exists as edges (u, c) in the session S and the set of prior voters is denoted as $U = \{u \mid (u, c) \in E_S\}$. The history $H = (V_H, E_H, w_H)$ is defined as the directed signed graph containing the votes from sessions chronologically prior to S.

We define a set of auxiliary graphs $A = \{G_1, G_2, \ldots, G_l\}$ based on external non-voting data. These auxiliary graphs can be either directed or undirected, weighted or unweighted, signed or unsigned. This is similar to the relational layer in *Multidimensional Social Networks* (MSN) described by Kazienko et al. [30] and Jankowski-Lorek et al. [28]. However, the auxiliary graphs capture different relations between a *subset* of the voting members that need not overlap. These relations will be used to generate additional

features for the vote prediction task.

Algorithm 1 describes how to generate a feature vector $\mathbf{a}$ from the auxiliary graphs A. The algorithm finds the intersection of the prior voters U and the neighbourhood of the current voter v in the graph G_i which we call the *voting neighbourhood*. Then the feature $\mathbf{a}_i$ is computed as the weighted sum of the voting neighbourhood plus the edge weight to the candidate (v, c) in the auxiliary graph G_i. Figure 3.1 provides an example with three auxiliary graphs and two prior voters u_1 and u_2. The dashed red edges are the votes cast in the current session S by the prior voters. We see, in the example, G_1 is a directed graph, G_2 is an undirected graph and G_3 is a signed directed graph. The current voter v has different relations to his voting neighbourhood in each auxiliary graph and therefore each feature is a different combination of edge weights from those graphs.

In addition to the auxiliary feature vector $\mathbf{a}$, we can also create triad features based on the historical voting graph H. Similar to Leskovec et al. [36] and Karimi et al. [29] we can form a set of unique triads T. Then, for each node u in the common neighbourhood of N_{vc} we can count the triad formed from the three vertices. Algorithm 2 describes this procedure.

We now create a feature matrix $\mathbf{X}$ for all the sessions in a given dataset. In the feature matrix, each row is the concatenation of the auxiliary feature vector and the triadic feature vector $\mathbf{x} = [\mathbf{a}, \mathbf{t}]$. The target vector $\mathbf{y}$ is the vector of true votes. Now, we train a linear machine learning model and use it to predict the votes in a test dataset.

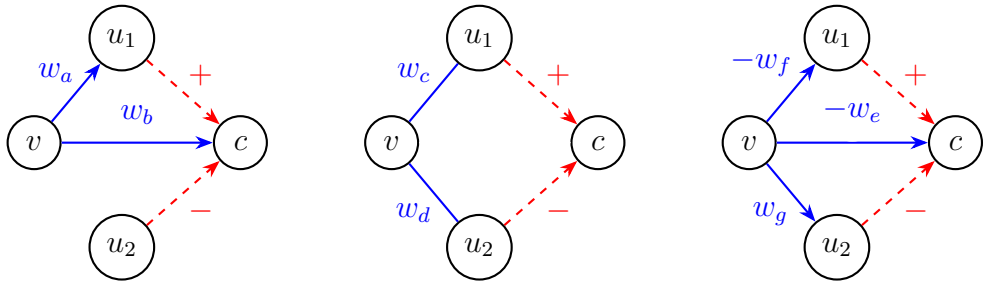


Figure 3.1: Example auxiliary features for v from combination of three graphs and two prior voters u_1 and u_2. Dashed red lines are prior votes in the session. Solid blue lines are edge weights in auxiliary graph. $\mathbf{a}_i$ is feature for voter v from auxiliary graph G_i

Algorithm 1: Auxiliary feature vector for voter v

Input: Voter v, Candidate c, Set of auxiliary graphs A, Current voting session S and Prior voters U
Result: Auxiliary Feature vector $\mathbf{a}$

Initialize $\mathbf{a}$ of length $|A|$
foreach G_i *in* A **do**
  $Z = N_i(v) \cap U$ `// neighbours in` G_i `who have voted`
  $\mathbf{a}_i \leftarrow 0$
  foreach z *in* Z **do**
    `/* vote in current session multiplied by the edge weight in` G_i `*/`
    $\mathbf{a}_i \mathrel{+}= w_S((z, c)) \cdot w_i((v, z))$
  end
  $\mathbf{a}_i \mathrel{+}= w_i((v, c))$ `// Add edge weight to candidate`
end
return a

Algorithm 2: Triad feature vector for voter v

Input: Voter v, Candidate c, Set of unique triads T, Voting history graph H
Result: Triad Feature vector $\mathbf{t}$

$k \leftarrow |T|$
Initialize counters $cnt_1, \ldots, cnt_k$ to 0
$N_{vc} = N_H(v) \cap N_H(c)$ `// common neighbours in` H
foreach u *in* N_{vc} **do**
  Let $\triangle$ be the triad formed by vertices $\{v, u, c\}$
  Classify $\triangle$ as the jth triad in T
  Increment counter cnt_j
end
$\mathbf{t} \leftarrow [cnt_1, \ldots, cnt_k]$
return t

3.4 Local Signed Network

In this section, we present an unsupervised method that can be used iteratively to predict the votes in a session. This method builds on top the concept of a *voting neighbourhood*, introduced in the previous section and relies solely on the social theories of balance and status in signed networks

to predict a vote. However, unlike the triadic features used in the supervised method, we propose to utilize generic measures of a network's adherence to the theories of balance or status. Then, we predict the vote that preserves these measures the best.

As defined in the previous section, the directed signed graph of the current session is $S = (V_S, E_S, w_S)$ where, v is the voter, c is the candidate and U is the set of prior voters in the session. $H = (V_H, E_H, w_H)$ is the directed signed graph that contains the historical voting records prior to the current session.

The first step is to construct a signed *relationship graph* $R = (V_R, E_R, w_R)$ from the historical voting graph H. The edges of this graph capture simple signed relationships between the voters such as *agreement* or *concurrence*. The relationship graph can also be constructed uniquely based on the details of the domain in which the voting occurs. Based on whether we use balance or status theory to predict the vote, the relationship graph is either unweighted or weighted respectively.

We now define the *Local Signed Network* (LSN) of a voter v as $LSN = S \cap_l R$. Where, $\cap_l$ is the *local intersection* of two graphs. Given two graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the local intersection is defined as seen in Equation (3.1).

$$G_1 \cap_l G_2 = (V_1 \cap V_2, E_1 \cup E_2). \tag{3.1}$$

Therefore, the LSN is the subgraph induced by the neighbourhood of v in R who have voted in the session S. We assume that candidate c is present in the relationship graph so that we get all the prior vote edges (u_j, c) in the LSN. The *voting neighbourhood* that we described in the previous section, has the same set of vertices as that of the LSN, i.e., $V_{LSN} = V_S \cap V_R$. There are three main types of edges present in the LSN. The first are the prior vote edges (u_j, c), from the prior voters to the candidate. The second are the relationship edges (v, u_j), from the voter to the prior voters. The third is the relationships between the prior voters (u_j, u_k). All these three types of edges can be seen in the undirected LSN shown in Figure 3.2c. We will now explain how to predict the edge (v, c) in the LSN using balance and status theories.

3.4.1 Prediction Based on Balance Theory

As described in Section 2.2.1, balance theory is applied to undirected signed graphs. Therefore, we construct an undirected signed relationship graph R using the voting history. The edge set E_R, represents the signed relations between nodes and should be symmetric in nature. For example, the proba-

bility of agreements or disagreement between a pair of nodes u and v. Now, we create the LSN from the intersection of the session graph S and relationship graph R. To keep the LSN an undirected graph, we ignore the direction of the voting edges (u_j, c).

Now, we turn to the task of predicting the sign of the edge (v, c) in the LSN. The voter v can cast either a positive or negative vote for the candidate c. This gives us two possibilities $w_{LSN}((v, c)) = +1$ or $w_{LSN}((v, c)) = -1$. We state that the voter aims to maintain the balance in the LSN. Therefore, we can predict the vote that when added as the edge (v, c) to the LSN results in a more balanced network. This indicates that we require a measure of the *imbalance* of a network. If $\overline{L}_{LSN}$ is the signed Laplacian of the LSN with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$, then Li et al. [40] show that the smallest eigenvalue λ_1 is a measure of the imbalance of the LSN. Therefore, the larger λ_1 is, the more imbalanced the network is and if $\lambda_1 = 0$, then the network is perfectly balanced.

Combining these concepts we have Algorithm 3. When given a LSN and an edge (v, c) to predict, we first assume the vote is positive, add it to the LSN and compute the smallest eigenvalue denoted by λ_1^+. Next, we assume the vote is negative and add the edge to the LSN and compute the corresponding smallest eigenvalue denoted by λ_1^-.

Now, if $\lambda_1^+ < \lambda_1^-$, then we can predict that the vote is positive, as the resulting LSN is more balanced. Similarly, if $\lambda_1^- < \lambda_1^+$, we predict a negative vote, as it results in a more balanced network. This deterministic rule does not capture the gap between λ_1^+ and λ_1^-. Therefore, the ratio $r = \frac{\lambda_1^-}{\lambda_1^+}$ can be used to express the confidence in predicting the vote is positive. As λ_1^+ approaches 0 (or λ_1^- increases), r approaches ∞ and when λ_1^- approaches 0 (or λ_1^+ increases), r approaches 0. Also, when $\lambda_1^+ = \lambda_1^-$ then $r = 1$, which indicates that we are equivocal between the vote being positive and negative. We convert the ratio r into a measure of the probability that the given edge (v, c) is positive by defining $p = 1/(1 + e^{(1-r)})$. Therefore, the output of the model is the probability that a vote is positive.

Figure 3.2 shows an example of how we can iteratively predict votes using balance theory. The current voter at every iteration i is $v = u_{i+1}$. We start in the first iteration $i = 1$ with one prior voter u_1 who has voted negatively for the candidate c seen by the dotted red line in Figure 3.2a. The current voter v has a negative relationship with u_1 indicated by the solid blue line. Now, in this triad we know "v disagrees with u_1" and "u_1 disapproves of c". Using the intuition provided by balance theory, we can predict that the edge (v, c) is positive as it results in the triad being balanced. This result can be verified empirically by observing the values of smallest eigenvalue. We see

that $\lambda_1^+ = 0$ and $\lambda_1^- = 1$ and therefore, the positive vote probability is $p = 1$. Now, in the next iteration, v becomes the prior voter u_2 and we add the edge (u_2, c) with the true vote (in this example we assume it was the same as the prediction made) and we get the next voter as the new v.

In the second iteration $i = 2$, the new voter v has a positive relation with the prior voter u_1 as seen in Figure 3.2b. By preserving the relation between u_1 and u_2 from the previous iteration, we have larger cycles in the current LSN. Similar to the previous iteration, we observe the smallest eigenvalues of the LSN. We conclude that a negative vote leads to more balanced network and positive vote probability is now $p = 0$. Now, in the third iteration $i = 3$, the smallest eigenvalue of the LSN in both cases are equal. Therefore, we are equivocal in the vote being positive or negative. Hence, the positive vote probability is $p = 0.5$.

Algorithm 3: Predict positive vote probability using balance theory

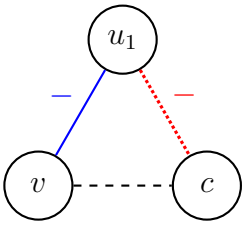
Input: Voter v, Candidate c, Local Signed Network LSN
Result: Probablity of edge (v, c) being positive

1 $w_{LSN}((v,c)) \leftarrow +1$ `// Assume positive vote`
2 Compute signed Laplacian $\overline{L}_+$
3 $\lambda_1^+ \leftarrow$ smallest eigenvalue of $\overline{L}_+$
4 $w_{LSN}((v,c)) \leftarrow -1$ `// Assume negative vote`
5 Compute signed Laplacian $\overline{L}_-$
6 $\lambda_1^- \leftarrow$ smallest eigenvalue of $\overline{L}_-$
7 $w_{LSN}((v,c)) \leftarrow 0$ `// Reset edge weight`
8 $r \leftarrow \lambda_1^-/\lambda_1^+$
9 $p \leftarrow 1/(1 + e^{(1-r)})$
10 **return** p

3.4.2 Prediction Based on Status Theory

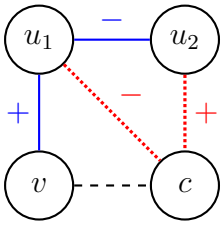
We mentioned in Section 2.2.2 that status theory is defined for directed signed networks. The relationship graph R should, therefore, be constructed from the history H as a directed signed network. The directed edges (u, v) should denote asymmetric relation between the nodes. For example, the ratio of *concurrence* which is a measure of times that u has voted after v in a session and agreed. The LSN created from the intersection of the session graph S and the relationship graph R is also a directed signed graph.

Similar to predicting the vote using balance theory, the vote can either be positive or negative. Now, we state that the voter aims to maintain



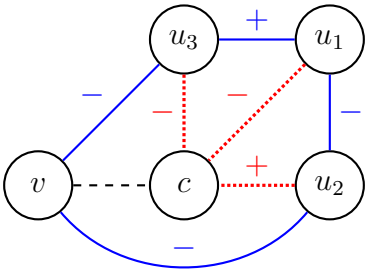
$\lambda_1^+ = 0, \lambda_1^- = 1$

(a) $i = 1$



$\lambda_1^+ = 0.58, \lambda_1^- = 0$

(b) $i = 2$



$\lambda_1^+ = 0.55, \lambda_1^- = 0.55$

(c) $i = 3$

Figure 3.2: Local Signed Network prediction based on balance theory. At every iteration i the dotted red lines are prior votes, solid blue lines are relationships based on voting records and the dashed black edge (v, c) is the edge whose sign is being predicted. λ_1^+ and λ_1^- correspond to the smallest eigenvalue of the signed Laplacian $\overline{L}_i$ based on whether $w((v, c)) = +1$ or $w((v, c)) = -1$.

the status in the resulting LSN. Therefore, we predict the vote (v, c) that when added to the LSN best preserves the status of the network. Therefore, we require a measure of how much a given network adheres to the theory of status. However, to the best of our knowledge, there are no existing methods to quantify and measure the extent that a network conforms to status theory. In this thesis, we present a quantitative definition of status theory in a network and also present a novel method of using the agony of a directed network to measure the compliance with status theory.

As we discussed in Section 2.2.2, signed edges can be interpreted as recognition of relative status. This means that a positive edge $u \xrightarrow{+} v$ indicates

that v has a higher status than u, and a negative edge $u \xrightarrow{-} v$ indicates that v has lower status than u. Let us assume that there exists an *implicit* status function $\sigma : V \to \mathbb{N}$ that maps each node in the network to a quantity that correlates with status. Therefore, the edges $u \xrightarrow{+} v$ and $u \xrightarrow{-} v$ indicate $\sigma(u) < \sigma(v)$ and $\sigma(u) > \sigma(v)$ respectively. We can use this relative ranking of status to predict the missing sign of an edge in a triad. Leskovec et al. [37] provide a way to compute this status function σ from the node degrees as shown in Section 2.2.2. However, this measure is still defined locally and can be used only to predict the sign of an edge. We require a framework to define violations to status theory and measure how much a network on a whole complies with status theory. We define that an edge $e = (u, v)$ *violates* status theory if $w(e)\sigma(u) \not\leq w(e)\sigma(v)$. Now, we can define when a network is perfectly complaint with status.

Definition 1. *Perfect Status Compliance A directed signed network $G = (V, E, w)$ with an implicit status function $\sigma : V \to \mathbb{N}$ is said to be perfectly status complaint if $\forall e = (u, v) \in E$, $\text{sgn}(w(e))\sigma(u) \leq \text{sgn}(w(e))\sigma(v)$.*

Therefore, the number of edges that are in violation to status theory can be a rudimentary measure of the status compliance of a given network. However, in the case of most signed directed networks, we do not possess the implicit status function σ. Hence, even computing the number of edges that violate status theory is not possible. However, if we can infer the status function σ from the structure of the signed network then we can measure the status compliance of that network.

We now use the concept of *agony* described in Section 2.4 to find the optimal hierarchy of a given directed network. However, agony and hierarchy were primarily defined for unsigned networks. We can easily remedy this by using the fact that a negative edge $u \xrightarrow{-} v$ can be transformed into a positive edge $u \xleftarrow{+} v$. If G is a signed network then we denote the unsigned directed network obtained from the transformation described as G'.

The notion of hierarchy is strongly related to status theory for signed networks. For instance, if G' is a DAG, then there are no cycles and we can find a status function σ, such that there are no edges that violate status theory. Therefore, let us consider that the rank function $r : V \to \mathbb{N}$ used to define agony is the status function σ for the signed network. Then, the agony of an edge in G' is a measure of the violation of that edge with respect to status theory.

The agony of an edge (u, v) in G' given the status function σ is defined as $\max(\sigma(u) - \sigma(v) + 1, 0)$. Therefore, agony is a quantification of the status violation of an edge. The agony of the network with respect to a status

function σ is defined as the sum of the agony of each edge in the network denoted by $A(G', \sigma)$. The agony of the network $A(G') = \min_\sigma(A(G', \sigma))$. is the smallest agony over all possible ranking of the nodes. In this way, the agony of the network is a more generalized measure of status compliance than just counting the number of violating edges. Therefore, we can use one of the algorithms mentioned in Section 2.4 to compute agony of G'. We call this value as the agony of the original signed network G and denote $\alpha = \alpha(G) = A(G')$. The complete process is outlined in Algorithm 4. Now, the agony of a signed network G is a measure of how far G is from being perfectly status complaint. Theorem 1 shows that when $\alpha = 0$ the network is perfectly status complaint.

Theorem 1. *Let $G = (V, E, w)$ be a directed signed graph. Then $\alpha(G) = 0$ if and only if G is perfectly status complaint.*

Proof. The transformed unsigned directed network is $G' = (V', E', w')$. The agony of a the directed network G' is 0 when the network has perfect hierarchy [23]. Therefore, there exists a status function σ such that the agony of all edges G', i.e., $\sigma(u) \leq \sigma(v), \forall (u, v) \in E'$. Therefore, there are no edges in G that violate status theory and G is perfectly status complaint. Similarly, when G is perfectly status compliant, we can find a status function $\sigma : V \to \mathbb{N}$, such that the agony of each edge in G' is 0. Therefore, the agony of the entire graph G' is 0, and correspondingly $\alpha(G) = 0$. Hence proved. □

Now, we can compute the agony of the LSN and utilize it to predict the sign of the vote. We follow a process similar to predicting the sign using balance theorem. First, first assume the vote is positive and add the edge (v, c) to the LSN and compute the agony and denote it α^+. Next, we assume the vote is negative, add the corresponding edge to the LSN and compute the agony and denote it α^-. If $\alpha^+ < \alpha^-$, then it means that the positive vote results in a LSN that has fewer violations of status and therefore, we can predict the vote is positive. Similarly, we predict a negative vote if $\alpha^- < \alpha^+$. Alternatively, we can also compute the probability of a positive vote as $p = 1/(1 + e^{(1-r)})$, where $r = \alpha^-/\alpha^+$. This process is detailed in Algorithm 5.

We see a directed LSN similar to the one in Figure 3.2a in Figure 3.3. The branches indicate the two possibilities for the vote edge (v, c). The left branch assumes that the vote is positive and adds it to the LSN. When we transform the negative edges in the LSN we get a cycle. As a cycle indicates that there is no hierarchy in the LSN, the agony $\alpha^+ = 3$ reflects the fact that each edge violates status theory. The right branch assumes that the vote is negative and includes it in the LSN. After transformation, we see that all the

Algorithm 4: Compute Agony for a directed signed network

Input: Directed siged graph $G = (V, E, w)$
Result: Signed Agony α of G

Initialize $G' = (V', E', w')$
$V' \leftarrow V$
foreach $e \in E$ **do**
  if $w(e) < 0$ **then**
    $e' \leftarrow (\text{dest}(e), \text{src}(e))$ `// Change direction of the edge`
    $E' \leftarrow E' \cup \{e'\}$
    $w'(e') \leftarrow -w(e)$ `// Make the weight positive`
  else
    $E' \leftarrow E' \cup \{e\}$
    $w'(e) \leftarrow w(e)$
  end
end
$\alpha \leftarrow Agony(G')$
return α

edges comply with status theory and therefore, the agony $\alpha^- = 0$. Now, the corresponding positive vote probability is $p = 0$, indicating that we would predict a negative vote. Note, this result is contradictory to the prediction made by balance theory for the same LSN.

Algorithm 5: Predict positive vote probability using status theory

Input: Voter v, Candidate c, Local Signed Network LSN
Result: Probablity of edge (v, c) being positive

$w_{LSN}((v, c)) \leftarrow +1$ `// Assume positive vote`
$\alpha^+ \leftarrow SignedAgony(LSN)$
$w_{LSN}((v, c)) \leftarrow -1$ `// Assume negative vote`
$\alpha^- \leftarrow SignedAgony(LSN)$
$w_{LSN}((v, c)) \leftarrow 0$ `// Reset edge weight`
$r \leftarrow \alpha^-/\alpha^+$
$p \leftarrow 1/(1 + e^{(1-r)})$
return p

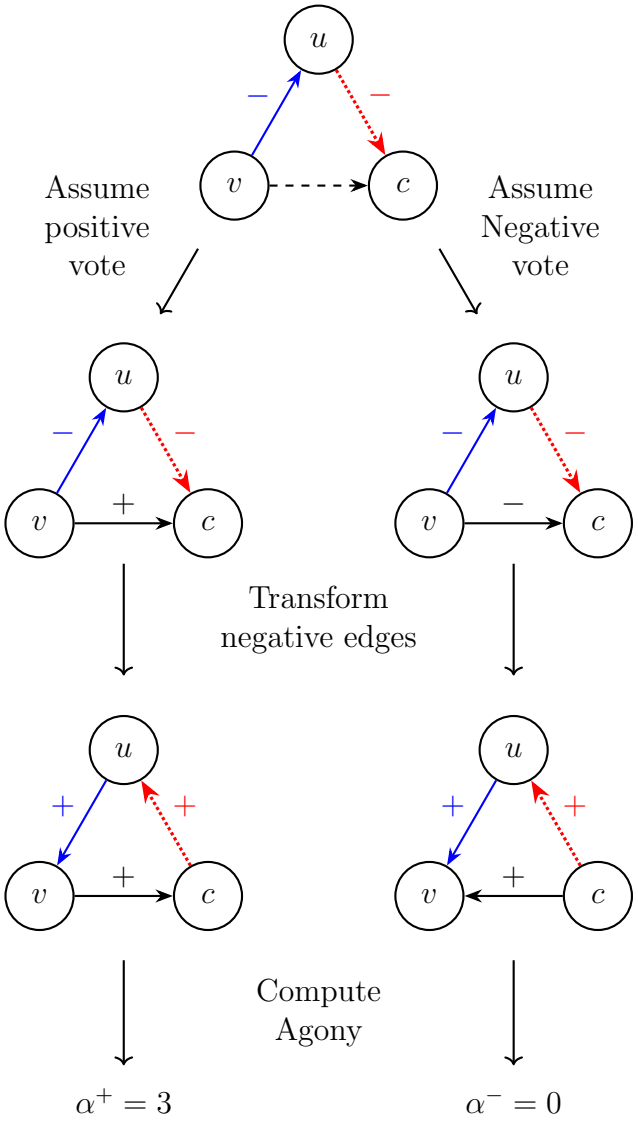


Figure 3.3: Example of LSN sign prediction using status theory.

3.4.3 Iterative Prediction Model

Now, Algorithm 6 describes a model that can predict the votes in a session iteratively. We assume we have the order of votes in the session denoted by O and the true votes function w^*. Then, we create the session graph and initialize it with the candidate c and the first voter as seen in line 6. This is because, we require information from at least one prior vote to effectively predict any vote. In most settings, such as bills in a parliament or promotion of a member in a community, there are always sponsors or nominators who

propose the candidate in that session. Therefore, we are justified in starting the session with the candidate and the first vote cast in the session graph S, which we assume is the nominator's vote. Next, for each subsequent voter v in the list, we add them to the session and create the LSN. The *Predict* function, in line 12 can be based on either balance theory (Algorithm 3) or status theory (Algorithm 5). After predicting the vote, we add the true vote of voter v to the session G and move on to the next voter in the list. After all the votes are predicted in the session, we can update the relationship graph H with the data from the current session S. This can include operations such as adding nodes that were not present in R and updating the edge weights based on the votes cast in the current session.

In this process we can predict all the votes in all the sessions by starting with an empty relationship graph and updating it after every session. Therefore, the model can be compared to a "*batch*" machine learning model, where each batch is a voting session and the model parameter is the relationship graph R. In a batch, a machine learning model will predict an outcome based on its parameters. Analogously, in a session, our iterative model predicts the vote using the the information gathered from the previous sessions contained in the relationship graph R. After a batch is complete, the machine learning model updates its parameters based on the data in the batch. Similarly, after the end of a session, our iterative model updates the relationships graph with voting data from the current session.

Furthermore, we can bootstrap the iterative model from a complete *blank slate* where R is an empty graph and then iteratively predicts sessions and updates R. Therefore, the model can improve after each session by incorporating the voting information into the relationship graph.

Algorithm 6: Iterative Prediction Model

Input: Candidate c, Relationship graph $R = (V_R, E_R, w_R)$, Order of voters in current session O and true votes w^*

Result: Predictions for current session

$k \leftarrow |O|$

$u \leftarrow O[1]$ `// First voter`

$V_S \leftarrow \{c, u\}$ `// candidate and first voter`

$E_S \leftarrow \{(u, c)\}$ `// first vote`

$w_S((u, c)) \leftarrow w^*((u, c))$ `// Assign true vote`

Initialize session graph $S = \{V_S, E_S, w_S\}$

$predictions \leftarrow \emptyset$

for $i \leftarrow 2$ **to** k **do**

  $v \leftarrow O[i]$

  $V_S \leftarrow V_S \cup \{v\}$

  $LSN \leftarrow S \cap_l R$

  $p \leftarrow Predict(v, c, LSN)$

  $predictions \leftarrow predictions \cup p$

  $E_S \leftarrow E_S \cup \{(v, c)\}$

  $w_S((v, c)) \leftarrow w^*((v, c))$ `// Assign true vote`

end

$Update(R, S)$ `// Update Relationship graph`

return *predictions*

Chapter 4

Wikipedia

In this chapter, we provide an overview of the inner workings and decision making processes of Wikipedia. Firstly, in Section 4.1 we state the fundamental principles of Wikipedia and how it guides editors on the website. Next, we describe the roles and responsibilities of the different categories of Wikipedia users in Section 4.2. Lastly, we explain the election process for administrators and discuss the voting behaviour in terms of existing research.

4.1 Principles of Wikipedia

Wikipedia is the largest online encyclopedia, with over six million pages of content in the English version. It is maintained by an open collaborative effort of multiple editors from all across the world. All the knowledge and content is free and is supported by the non-profit Wikimedia Foundation. The size and popularity of Wikipedia is attributed to the ability for anyone to edit any content. All editors on Wikipedia follow five fundamental principles, called the "Five Pillars", shown in Figure 4.1. These five pillars provide a foundational framework for collaboration amongst users and contribution towards the betterment of the Wikipedia project.

The first pillar states that Wikipedia is first and foremost an encyclopedia [63]. Therefore, it must not contain any original research, propaganda or advertisements [61]. Materials that do not have reliable references will be removed by other edits.

The second pillar specifies that articles on Wikipedia should strive for a neutral point of view. This might include presenting multiple perspectives on the same subject accurately and not championing any one viewpoint as "correct" or "the truth". If disagreements are present, then discussions must take place for building consensus.

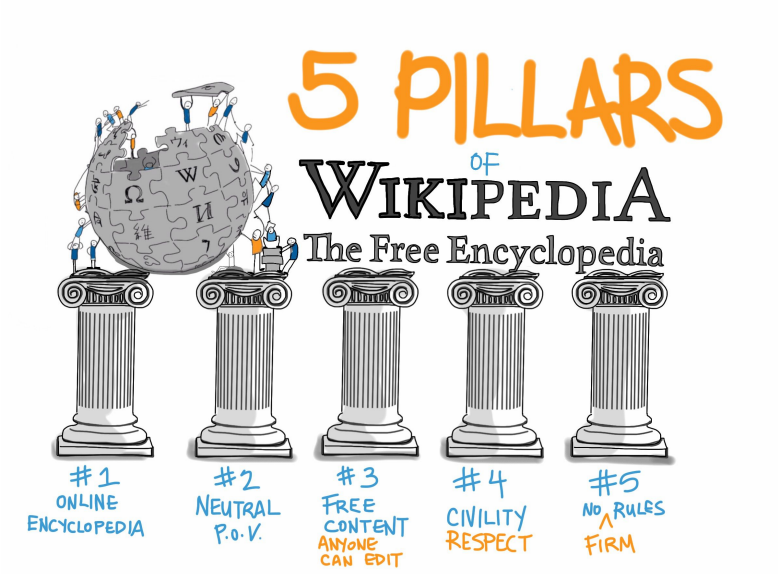


Figure 4.1: Five Pillars of Wikipedia. Image downloaded from `https://www.flickr.com/photos/gforsythe/21684596874`

The third principle enshrines the ideal that all content available on Wikipedia is free to edit and share. However, this does not mean copyright violation and plagiarism is tolerated by the community. There is no ownership of an article by an editor; anyone may freely modify any content.

The fourth pillar describes Wikipedia's code of conduct. It asks users to act in good faith and assume good faith on the part of other editors. Wikipedia etiquette urges disputes and disagreements, such as edit wars [58], to be resolved with civility while respecting other editors.

The fifth and last pillar reminds users that all rules in Wikipedia are essentially just policies and guidelines meant to help with collaboration. They can evolve and change to reflect the requirement of the community. It assuages the fear of making mistakes and encourages editors to be bold, though not reckless.

4.2 Formal Organization of Wikipedia

In this section, we describe the various categories of users and explain their roles and responsibilities. All the facts and figures we provide in this thesis are from the English version of Wikipedia. We define a "user" of Wikipedia as a person who contributes to the encyclopedia and a "reader" as someone who simply accesses the content.

Wikipedia began as a completely open platform with no restrictions on who could edit a page or create a new article. Changes and edits that were made to a page were published immediately. This led to many pages that contained erroneous or nonsensical text, or biased content. Therefore, this led to the English version of Wikipedia introducing restrictions and tools to protect the more controversial pages. They also introduced categories of users to help protect and maintain the quality of the content available on Wikipedia. We proceed to explain the four main user types as seen in Figure 4.2.

(a) Editors

(b) Administrators

(c) Bureaucrats

(d) Arbitration Committee

Figure 4.2: Logos for each category of user that signify the role that they play in the Wikipedia community.

4.2.1 Editors

Editors (or Wikipedians) are the primary users who edit and create all the content on Wikipedia [64]. Figuratively, they hold Wikipedia in the palm of their hands, as seen by the logo in Figure 4.2a. There are two main types of editors on Wikipedia, namely *registered* and *unregistered*. A registered user is someone who has a unique username and a permanent *talk page* to communicate with other users. By contrast, unregistered users contribute without a registered username and are usually referred to as *IPs*, as they are only identified by their IP addresses. Unregistered users usually have similar

rights as those of regular users to edit, discuss and contribute, but with certain exceptions [59]. Unregistered users cannot create a new article, edit a protected page, become administrators or vote in elections to promote users within Wikipedia. As the focus of the thesis will be on the elections within Wikipedia, when we refer to editors in the coming chapters and sections, we refer to registered users.

Wikipedia has over 38 million registered users, and this number is constantly rising. However, only roughly 0.37% ($\approx$ 144000) of registered users are active, i.e., have performed some action in the past 30 days. An even smaller percentage of those active users participate in the community discussion forums on Wikipedia. Now, we will explain what tasks editors perform and how contributions are recorded in Wikipedia.

Each page in Wikipedia is classified into a *namespace* based on the type of information that page contains [60]. Namespaces separate pages into sets to distinguish content pages from administrative or editor related pages. For example, the MAIN (or ARTICLE) namespace contains all the encyclopedic content and the USER namespace contains the user pages and information related to their user accounts. Each page in Wikipedia also has a corresponding *talk page*, which are used by editors to discuss changes to the page in question. For instance, the USER TALK namespace has talk pages corresponding to each user page and acts as a system to message particular users. Figure 4.3 shows a list of the subject namespaces and their corresponding talk namespaces. Now, we define a *user contribution* as any addition, deletion or modification of a page under any namespace in Wikipedia [55]. Wikipedia collects and stores every user contribution so that it can track cases of vandalism and copyright infringement.

The quality and quantity of the contribution of each editor varies significantly. There are many occasional users who merely correct minor spelling and grammar errors in articles. At the same time, there are dedicated editors who constantly create new articles, update large portions of text, and include new references and images.

4.2.2 Administrators

Administrators (or *admins*) are editors who are given access to certain tools and powers to maintain content on Wikipedia. Administrators can delete and restore deleted pages, block and unblock users and IP addresses from editing, and protect and remove protection from sensitive pages [56]. These tools are associated with a mop that is used to clean up Wikipedia and is represented by their logo, seen in Figure 4.2b. There are currently 1141 administrators, of whom 500 are active. Although admins have access to these tools, they

Wikipedia data structure

Namespaces			
Subject namespaces		Talk namespaces	
0	(Main/Article)	Talk	1
2	User	User talk	3
4	Wikipedia	Wikipedia talk	5
6	File	File talk	7
8	MediaWiki	MediaWiki talk	9
10	Template	Template talk	11
12	Help	Help talk	13
14	Category	Category talk	15

Figure 4.3: A list of the namespaces in Wikipedia [60].

are considered to be no more important than regular editors. Administrators are elected through a week-long process called *Request for Adminship* (RfA), at the end of which successful candidates are instated by a Bureaucrat. We will cover the RfA process in detail in the coming sections.

Along with the tools and power, administrators also have certain responsibilities. They are not to misuse the tools at their disposal in conflicts of interest or disrupt Wikipedia by acting in bad faith. Administrators serve indefinitely, but can be removed by Bureaucrats on the decision of the Arbitration Committee for abuse of powers or inactivity. Admins help with various areas of Wikipedia, such as processing administrative backlogs, helping with ant-vandalism efforts, and managing copyright issues.

4.2.3 Bureaucrats

Bureaucrats (or *Crats*) are users who perform certain actions [57]. They are usually administrators and oversee procedural rules and enforce decisions. There are a total of 19 bureaucrats currently in the English Wikipedia. Bureaucrats are involved in the granting or revoking of administrator status to users and adding and removing bots (software robots that carry out repetitive tasks on Wikipedia). Bureaucrats are bound by the policy and the consensus of the community in granting these roles or permissions and are,

therefore, expected to be good arbiters of consensus. Hence, they should be able to identify criteria for a "consensus" and also explain theirs reasons behind their actions when requested.

Bureaucrats are also elected through a process similar to RfA, called *Request for Bureaucratship* (RfB), but higher thresholds of acceptance are usually demanded for considered selection. Interestingly, Bureaucrats are also appointed following the final decision of another Bureaucrat, therefore, they have complete control over the whole process. However, a Bureaucrat cannot revoke the bureaucratic position of others. They also carry out the requests from the Arbitration Committee to remove the permissions and privileges of admins or bots. As their name suggests, Crats perform only bureaucratic duties and are therefore represented by the logo seen in Figure 4.2c.

4.2.4 Arbitration Committee

The *Arbitration Committee* (or *ArbCom*) resolves disputes that have not reached a resolution through community discussion or administrator oversight [54]. Their goal is to decisively bring binding solutions to ongoing disputes and is reflected in the fact their logo is a balance scale, as seen in Figure 4.2d. It is formed by a panel of experienced editors, usually administrators, who are elected by the community annually. There are currently 11 active members of the ArbCom.

The ArbCom only deals with disputes related to editor conduct and not content related disputes. It can impose sanctions that would restrict editors from contributing to certain topics and also recommend the revoking of administrative privileges in cases of misuse. Although the ArbCom can take the initiative on matters it deems are important, it usually acts on formal requests made to the committee. As it is the last step in dispute resolution, it only accepts a case when all other methods have failed. This is evident from the fact that only 9 cases were accepted in 2019.

4.3 Request for Adminship

In this section, we will describe the election process to select administrators in the English version of Wikipedia called *Request for Adminship* (RfA). We cover the origin and history of the process, the evolution of the format and the properties that lead to successful candidates. Lastly, we also cover the existing research that has been carried out in understanding and predicting RfAs.

In the early days of Wikipedia, the founder, Jimmy Wales, directly sent emails to the users to appoint them as administrators. Jimmy said that he felt that getting administrative privileges is "not a big deal". However, as the Wikipedia community grew, a long and intense process was developed to select future administrators. A RfA is a week-long period during which all registered Wikipedia users can vote on a candidate standing for the position of administrator.

There are four main phases of a RfA: the nomination and beginning the period, answering questions posed by the community, voting to show support, opposition or neutrality towards the candidate, and the closing of the RfA by a Bureaucrat.

The first phase begins with the creation of a RfA page for the nominee. The candidates are most often nominated by another well-known and respected editor. However, self-nomination is a possibility. Self-nominated candidates are usually under more scrutiny to ensure they are neither overeager new users nor editors with prior issues. Nominations are usually accompanied by an introductory statement from the nominator indicating the qualities the candidate possesses. Nominees can decline a nomination if they wish to, in which case the RfA is closed immediately as unsuccessful. Therefore, nominators usually only choose candidates who show good promise and discuss the potential nomination prior to starting the RfA process.

Once a candidate accepts the RfA nomination, they are required to answer three standard questions.

1. What admin work do you intend to take part in?
2. What are your best contributions to Wikipedia, and why?
3. Have you been in any conflicts over editing in the past or have other users caused you stress? How have you dealt with it and how will you deal with it in the future?

The first question aims to discern the value addition that a particular candidate will bring to Wikipedia if given admin privileges. The community tends to look for initiative from nominees in utilizing existing tools to help with chores such as reverting errors, identifying vandalism or copyright infringements. The answer to the second question provides the community the candidate's achievements and quality of work. Editors who have several multiple good contributions tend to be more successful. In answering the third question, candidates demonstrate their conflict management skills. The community values users who can interact in a civil manner. As an administrator is involved in resolving disputes, users who were involved in heated discussions or edit wars are generally unfavourable. Apart from these three fixed

questions the candidate may also receive several open questions aimed at testing their knowledge of Wikipedia procedures or gaining their opinions on controversial issues.

Once all questions have been answered, the RfA moves to the voting phase. During this phase, any registered user may vote in either the Support, Oppose or Neutral sections. Votes are generally followed by a comment providing reasoning that explains their vote. Candidates can reply to opposition comments to try and resolve any issues and convert their views. However, candidates should refrain from verbose rebuttals as it might invite more opposition. This phase is nerve-racking for the candidate as the tide of the election changes constantly throughout the week and it is not possible to reply to every comment in a civil and respectful manner.

At any point in the RfA, the candidate can withdraw their nomination for any reason. At the end of the week, a Bureaucrat halts the voting and proceeds to read all the comments. The Bureaucrat has to conclude if consensus has been reached or not regarding the nomination. Bureaucrats are highly experienced and will discount votes cast by sockpuppets (users who have multiple accounts) and meatpuppets (new users recruited to influence decisions). Although the decision is not based on majority voting, RfAs with more than 75% support generally pass and by contrast ones with lesser than 65% support are bound to fail.

The Bureaucrat can also invoke clauses such as "Not a snowball's chance in hell" (WP:SNOW [62]) and "Not Now" (WP:NOTNOW [53]) to terminate RfAs that they deem have no chance to succeed. These measure exists so that frivolous RfAs do not waste the time of the community. If the Bureaucrat decides that the nomination is successful, the candidate is promoted and the RfA is closed as successful. If the nomination fails then the Bureaucrat explains their reasoning and closes the RfA as a failure. Renomination of a failed candidate can occur after waiting for a reasonable period of time from the previous failed RfA.

RfAs have been extensively studied from a sociological and behavioural aspects [17, 33]. Burke et al. [9] proposed a model based on RfA guides to predict the success of a potential nomination. Since then, there have been various models based on social networks [10, 46, 47] or user features and contributions [5, 45] to identify influential voters and overall voting patterns.

Chapter 5

Implementation

In this chapter, we will outline the experiments carried out on the Wikipedia elections of administrators using the vote prediction models that we presented in chapter 3. Firstly, we describe the existing sources of data from Wikipedia and the datasets used in the experiments in Section 5.1. Next, in Section 5.2, we discuss the implementation of the linear combination of graphs model described in Section 3.3. Then, we cover the implementation of the vote prediction models based on the theories of balance and status in signed networks in Section 5.3. Furthermore, in Section 5.4, we discuss the experiments conducted on the voting order and its impact on the predictive power of the models proposed. Lastly, we explain the metrics which we can use to evaluate the performance of the models in Section 5.5.

All implementations and datasets can be found at `https://github.com/ananth1996/Wikipedia`

5.1 Datasets

As we discussed in Section 4.3 and 4.2.1, Wikipedia keeps detailed information on the election proceedings for the RfA process as well as contributions made by every editor on Wikipedia. These act as sources to get data regarding the elections and user contributions. There are existing datasets compiled by the Stanford Network Analysis Project (SNAP) [38] on both Wikipedia RfAs and edit histories. However, the RfA dataset has missing features and timestamps for votes that would restrict the usability in the proposed voting models. Similarly, the *wiki-meta* and *wiki-talk* datasets only possess information until 2008 and lack a username mapping to the network nodes. Due to these limitations, we proceeded to scrape Wikipedia dumps and APIs to obtain our own RfA and user contribution datasets, which we will now

describe.

5.1.1 Wikipedia RfA Data

To obtain the RfA data, we parsed through the entire XML dump of Wikipedia from January 2019. We filtered the pages related to the RfA process and then extracted each vote and the corresponding comment and timestamp. Each vote extracted has the features shown in Table 5.1.

Table 5.1: Features of each vote in the WIKI-RFA dataset

Feature	Data Type	Description
SRC	text	username of the source
TGT	text	username of the target
VOT	$[-1, 0, 1]$	Oppose, Neutral or Support vote
RES	$[-1, 1]$	Failure or Success of RfA
YEA	date	year of the RfA
DAT	date & time	timestamp of the vote
TXT	text	accompanying textual comment
UID	alphanumeric	unique identifier for the RfA

As we can see, the format of the data is very similar to the SNAP dataset. We have an additional unique identifier field, called UID, to aid in distinguishing RfA of users who have had multiple nominations. We collected 226781 votes from 4557 elections with over 13000 unique usernames. There are 166214 ($\approx 73\%$) support, 46918 ($\approx 20\%$) oppose and 13649 ($\approx 6\%$) neutral votes. As the voting format of RfA changes throughout the years, there were issues in successfully extracting the source username or timestamp information. Regardless, only 1.6% of votes have missing timestamps and 0.4% have a missing source. We will refer to this dataset as WIKI-RFA and it will provide the information regarding the votes cast in a RfA.

5.1.2 User Contribution Data

As we discussed in Section 4.2.1, every edit made by a user is stored as a contribution. Wikipedia provides an API to query all the contributions of a particular user [43]. We utilized this API and collected the contribution data of all the unique users we obtained from the WIKI-RFA dataset. There are 16 features that the API provides for each edit; we describe the most import features in Table 5.2.

Table 5.2: Important features of each contribution in the USER-CONTRIB dataset

Feature	Data Type	Description
user	text	username of the editor
title	text	title of the page edited
namespace	int	namespace of the page edited
timestamp	date & time	timestamp of the edit
size	int	new size of the edit
sizediff	int	size delta of the edit against its parent
new	boolean	if the editor created a new page
minor	boolean	if it is a minor edit
comment	text	accompanying comment

As many users change their username, some of the usernames present in the WIKI-RFA dataset might not have any contributions linked to their old usernames. We were able to collect the user contribution details of more than 11000 users, amounting to 100GB of data. We call this dataset USER-CONTRIB and it provides a wealth of information on the editing habits of the users who take part in Wikipedia RfAs. For instance, grouping the contributions of a particular user by the namespace, we get the proportion of the edits in different Wikipedia namespaces and the respective sizes and quality of their edits.

5.2 Graph Combination Model

In this section, we describe how we implemented the linear combination of graphs framework proposed in Section 3.3 for predicting votes in Wikipedia RfA elections. We call this the *Graph Combination* model. The model requires auxiliary graphs created from other non-election based information as well as triadic features extracted from the voting data. Firstly, we discuss the auxiliary graphs that we create from the USER-CONTRIB dataset. Next, we explain the nomenclature and collection of triadic features from the WIKI-RFA data. Then, we describe the process of preparing the data to suit the supervised machine learning task as well as preventing any potential data leaks. Lastly, we discuss the logistic regression model that we use as the linear classifier trained on the features derived from the auxiliary and signed networks.

The terms used in Chapter 3 can now be defined for the problem of

predicting votes in a Wikipedia RfA. A candidate c is the nominee who wishes to gain administrators privileges in the Wikipedia RfA. The voters v are the registered users in Wikipedia. A session relates to the proceedings of a particular RfA.

5.2.1 Graphs

First, we discuss the creation of the topic similarity network of users. Then, we describe the process of forming the talk graph between users. Lastly, we define the triadic features we extract from the previous voting data.

5.2.1.1 Topic Similarly Graph

In Table 5.2, we see that every contribution has a title of the page where the edit was made. The most edited page titles of a user help in understanding the topics they are interested in. Therefore, for a particular user, we gather all their edits in the MAIN namespace. We choose the MAIN namespace as it contains all the content articles on Wikipedia. By contrast, a user's edits in other namespaces such as, USER and HELP, are not indicative of the topics in which they possess knowledge. Then, we count the number of edits grouped by each page title and choose their top 100 most edited pages in the MAIN namespace. Then we create a set of the words from all the top 100 page titles and remove common stop words using a natural language corpus. This set now indicates the user's topics of interest. Once we have collected the topic set for all the unique users in the WIKI-RFA dataset, we can compute the similarity between a pair of users using the Jaccard similarity measure. Then, we can take this similarity measure and construct a undirected weighted graph where a link between any two nodes indicates the similarity in the topics of the corresponding users. However, we threshold the value of similarity so that we can obtain only meaningful edges and not a complete graph.

5.2.1.2 Talk and Interaction Graph

We discussed in the previous chapter how every registered user has a talk page and how it is used as a medium of communication. Therefore, we can gather the contributions of a certain editor in the USER TALK namespace and use it to measure their interactions with other users. We will create two auxiliary graphs in this manner. The first is a *user talk graph*, where each edge contains the number of times they have written on another user's page. The second is a *interaction graph*, in which an edge only indicates

if two users have interacted via a talk page. We can obtain the number of talk page edits and the target user by grouping by the page titles and extracting the username from the page title respectively. These graphs will be directed in nature and the talk graph is weighted, while the interaction graph is unweighted. In both these graphs, an edge $u \to v$ indicates that user u has written in the talk page of user v.

In the Line 3 of Algorithm 1, we compute the neighbourhood of a node v in graph G_i as N_i. We can define N_i in directed graphs G_i as only the successors of a node rather the union of successors and predecessors. This allow us to understand the influence of edge direction in directed auxiliary graphs. Therefore, we will construct two more additional auxiliary graphs which are *reversed*, i.e., an edge $u \to v$ indicates that user v has written on the talk page of user u. Hence, we can compare the benefit each direction brings to the model by analysing the feature importances of their respective auxiliary graphs.

5.2.1.3 Signed Graph and Triadic Features

The WIKI-RFA dataset contains the voting information of users in RfAs. These votes form a signed directed network. Therefore, we can utilize the triadic features framework as proposed by Leskovec et al. [36].

We utilize a slightly modified naming scheme to identify unique triads in the RfA data. Consider the we have a voter v, a candidate c and a third node u. Then, the edge we wish to predict is (v, c) and the other edges (v, u) and (u, c) form a triad. There are two directions for the edges (v, u) and (u, c) and each edge can have three values, namely -1, 0 or $+1$ corresponding to a oppose, neutral or support vote respectively. This leads to $2 \times 2 \times 3 \times 3 = 36$ possible triads.

We denote the edge $v \to u$ as "F" and the edge $v \leftarrow u$ as "B" indicating a forward or a backward edge respectively. Similarly, the edge $u \to c$ is "F" and $u \leftarrow c$ is "B". The edge labels are "$-$", "0", or "$+$" corresponding to a oppose, neutral or support vote. Therefore, using this nomenclature, the triad "FB$+-$" represents the edges $v \xrightarrow{+} c$ and $u \xleftarrow{-} c$. Figure 5.1 shows more examples of this triad nomenclature.

We store all the 36 unique triads in the set T and then utilize it to count the triads for a particular edge, as seen in Algorithm 2. Therefore, for each edge to be predicted (v, c), we have a triadic feature vector of length 36 containing the counts of the triads formed by all the common neighbours u.

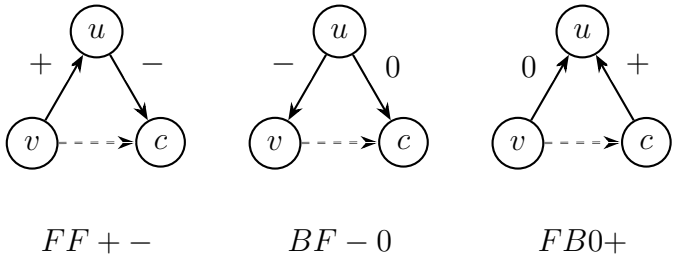


Figure 5.1: Examples of triad nomenclature in Wikipedia RfA elections. Dashed edges are votes to be predicted and solid edges are votes from previous RfAs.

5.2.2 Data Preparation

As only roughly 6% of all votes are neutral votes, we will not try to predict neutral votes. This is in line with the Wikipedia RfA process where neutral votes are not counted for the support percentage. However, we will use the neutral votes to gather the triadic features and can utilize the additional information to predict votes.

As we discussed in Section 3.2, the graph combination model is an extension of a sign prediction model for the task of predicting votes. A major requirement to predict votes is to ensure that there is no *data leakage* when creating the training features $\mathbf{X}$. A data leak is when we have information about the future available in the training data. This can cause the model that we train to overfit on the leaked data and not generalize. Kairimi et al. [29] outline a process to split a dataset chronologically and gather information respecting the boundary dates at the location of the splits. Similarly, for our problem setting, we divide the whole WIKI-RFA into three parts, namely *dev*, *train* and *test*. As we are predicting votes, we split the datasets based on the number of votes chronologically and round up to the closest RfA so that it is contiguous.

The *dev* (or development) dataset will be the set of RfAs which we use to construct the auxiliary and signed graphs. We ensure that the USER-CONTRIB dataset is also restricted to the edits that happened until the date of the last RfA in the dev dataset. Therefore, all the five auxiliary graphs and the signed graphs are created only with information that is present in the time frame of the dev dataset.

Next, the *train* (or training) dataset is what we use to create the feature matrix $\mathbf{X}$ and target matrix $\mathbf{y}$. In this dataset, we only consider the support and oppose votes to be part of the prediction task and hence, filter out all the neutral votes. Now, for each vote, we create the auxiliary feature vector

a using the five auxiliary graphs and the triadic feature vector **t** from the signed voting graph as described in Algorithms 1 and 2 respectively. These features are concatenated into a feature vector **x**, which is of length 41 (5 auxiliary and 36 triadic features) and is a row of the feature matrix. The corresponding true vote is also collected in the target y. The *dev* split ensure that the auxiliary and triadic features in *train* do not overlap with the *test* split. Therefore, this allows the feature matrix $\mathbf{X}$ to be independent of time and we can use methods such as k-fold to cross validate the model.

Lastly, the *test* dataset contains votes that the model trained on the training dataset would not have seen. This can be used to evaluate the performance of the model. The feature matrix for the test dataset, $\mathbf{X}_{test}$ and the target, $\mathbf{y}_{test}$ are also constructed in a similar manner. For each vote in the test dataset, we gather the auxiliary and triadic features from the same graphs as we used for the training phase. We create each row of $\mathbf{X}_{test}$ by concatenating these features and gathering the true votes as the target.

As the auxiliary features and triadic features have different ranges, we standardize both training and testing feature matrices so that all features have a mean of zero and standard deviation of one. This will help the linear models train better and reach an optimal solution faster as well as allow for ease in interpreting the coefficients of the linear model.

5.2.3 Supervised Classification

Once we have created the training and testing feature matrices, $\mathbf{X}$ and $\mathbf{X}_{test}$ and the target vectors, $\mathbf{y}$ and $\mathbf{y}_{test}$, the task is a regular supervised classification problem. We can use any traditional linear classification model such as support vector classifier (e.g.,linear SVC), logistic regression model or gradient boosting method (e.g., XGBoost). We choose a logistic regression (LR) model for its interpretability and robustness to overfitting.

Given a feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ with n features, a logistic regression model learn to predict the probability of the form

$$P(\text{support} \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \boldsymbol{\beta}\mathbf{x})}} \tag{5.1}$$

Where β_0 and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$ are the coefficients that the model learns using the training data.

The WIKI-RFA dataset has a class imbalance problem. Support votes are 73% compared to oppose votes at 20%. Therefore, we will utilize class weights inversely proportional to the class frequencies while training so that the model learns to predict negative votes effectively. As the training fea-

tures $\mathbf{X}$ are independent of time, we use k-fold cross validation to tune the regularization parameter for the logistic regression model.

5.3 Local Signed Network Models

We now discuss the implementation of the local signed network models discussed in Section 3.4 to predict votes in Wikipedia RfAs.

These models are iterative models and an important feature is that they are unsupervised. Therefore, they do not require any learning of parameters or preparation of data for training. Consequently, we can bootstrap the models to start from the first available RfA. We achieve this by beginning with an empty relationship graph R. In the first RfA, the LSN for all the votes contain only the nodes for voter v and candidate c. Therefore, the model will predict all votes with probability 0.5 of being support votes, as there is no information available. After the first RfA is over, the relationship graph R will be updated with the voting details. Now, in the second RfA there is more information present and the model can predict votes with more certainty. In this manner, the models can iteratively learn and predict all the votes present in the WIKI-RFA dataset.

In a similar fashion, the iterative models elegantly handles new users for whom we have no information . If at any point the current voter v is new and there is no information in the relationship graph , then the model predicts support vote probability of 0.5, because the LSN contains only the nodes v and c . This new voter is then integrated into the relationship graph when it is updated after the RfA session, shown in Line 17 of Algorithm 6. Therefore, this new voter's information is now available for future vote predictions.

In this thesis, we wish to separately study the votes that are predicted with no information. Therefore, in our implementation we specifically mark these votes. Consequently, we can accurately evaluate the iterative model using only the votes predicted with information. Then, we can analyse the distribution of the informationless votes and devise strategies to effectively guess the vote in the cases when a voter is new. Lastly, we can verify if there new voters follow some herd mentality when they vote for the first time.

We proposed two iterative models in Section 3.4, one using balance theory and another using status theory. Both these models make use of only the votes cast in sessions. Therefore, we will use the WIKI-RFA dataset for the iterative models. First, we describe the iterative balance model and define the relationship graph based on *agreement* between voters in Wikipedia RfAs. Secondly, we explain the iterative status model and the relationship graph based on the *follower ratio* in RfAs.

5.3.1 Iterative Balance Model

The *Iterative Balance Model* uses balance theory in the local signed network to predict the votes of independent voters in a Wikipedia RfA. As discussed in Section 3.4.1, we require a signed symmetric measure between two voters. We now propose a measure based on the *agreement ratio* between two users u and v. The ratio is the number of times u and v have voted similarly, divided by the number of common RfAs they have participated in. For example, if u and v have participated in 12 common RfAs and have voted the same in 9 RfAs, then the agreement ratio is 0.75. This indicates that they agree more than they disagree. Therefore, if a pair of users have an agreement ratio of 0.5, then they neither agree or disagree with each other. The agreement ratio is symmetric and we covert it into a signed measure by subtracting 0.5 from the ratio.

Hence, we define a signed undirected *agreement graph* $A = (V_A, E_A, w_A)$, where the weight function is defined as seen in Equation (5.2).

$$w_A((u,v)) = \frac{\text{Number of times } u \text{ and } v \text{ have voted similarly}}{\text{Number of common RfAs for } u \text{ and } v} - 0.5 \quad (5.2)$$

This agreement graph A, is the relationship graph R for the iterative balance model described in Algorithm 6. In Line 17 there is a a method, $Update(R, S)$ to update the relationship graph after the end of a voting session. Therefore, we require a method to update the signed weights in the agreement graph A given the RfA voting details in a session S.

For notational ease, we assume that each edge $e = (u, v) \in E_A$ contains two attributes, $e.agree$ and $e.common$, the agreement ratio and the number of common RfAs between the nodes respectively. Then, once we get the voting information from the session, we can update the agreement ratio and the number of common RfAs in a straightforward manner. This process is shown in Algorithm 7.

We can bootstrap the model by beginning with an empty agreement graph A. Then for votes with no information the predicted support probability is 0.5. After the RfA session is over, these new voters will be incorporated into the agreement graph. Therefore, from the next RfA, the model has information on the voters it has now incorporated.

5.3.2 Iterative Status Model

The *Iterative Status Model*, as described in Section 3.4.2, utilizes status theory in the LSN to predict votes. Therefore, to predict votes in Wikipedia

Algorithm 7: Update Agreement graph after a session

Input: Session graph S, Candidate c, Agreement graph A
Result: Updated Agreement graph A

```
// Get all voters
```
$O \leftarrow V_S - \{c\}$
Order O by timestamp
for $v \in O$ **do**
  $vote_v \leftarrow w_S((v, c))$
  foreach *u who voted after v* **do**
    $vote_u \leftarrow w_S((u, c))$
    $e \leftarrow (v, u)$
    if $e \in E_A$ **then**
      $agree \leftarrow e.agree$
      $common \leftarrow e.common$
      if $vote_v = vote_u$ **then**
        $agree \leftarrow ((agree \cdot common) + 1)/(common + 1)$
      else
        $agree \leftarrow (agree \cdot common)/(common + 1)$
      end
      $common \leftarrow common + 1$
    else if $vote_u = vote_v$ **then**
      `// if` e `is a new edge`
      $common \leftarrow$ number of elections v and u have in common
      $agree \leftarrow 1/common$
      $E_A \leftarrow E_A \cup \{e\}$
    end
    $e.agree \leftarrow agree$
    $e.common \leftarrow common$
    $w_A(e) \leftarrow e.agree - 0.5$
  end
end
return A

RfAs, we require a directed signed relationship graph. Similar to the agreement ratio for the iterative balance model, we propose a *follower ratio* and a corresponding directed signed *follow graph* $F = (V_F, E_F, w_F)$.

An edge $u \to v$ in F indicates that node u follows v in RfAs. In more detail, for the RfAs in which both user u and v have voted, u is said to follow v, if u votes after v and u votes the same as what v had voted. Note, it is not necessary for u to vote *immediately* after v, but just vote *chronologically* after v. Then, we define the *follower ratio* as the number of times u has agreed with v when u has voted after v, divided the total number of RfAs in which u has voted after v. For example, if u and v have 12 RfAs in common and in 8 of those, u has voted after v and in 5 out of 8, u has voted the same as v, then the follower ratio is $5/8 = 0.625$. Therefore, if the follower ratio is below 0.5, it indicates that u tends to vote the opposite of what v has voted. Also note that, if the follower ratio for (u, v) is 0.625, the follower ratio in the other direction (v, u) is not necessarily the same. Therefore, it is not symmetric and we can convert it into a signed measure by subtracting 0.5 from the follower ratio. The weight function w_F for the follow graph can be defined as seen in Equation (5.3).

$$w_F((u, v)) = \frac{\text{Number of times } u \text{ voted after and agreed with } v}{\text{Number of times } u \text{ voted after } v} - 0.5 \quad (5.3)$$

When we create the LSN, we only consider the edges of type $v \to u_i$ from the follow graph F. This is because the voter v is voting after the voters in U. Therefore, the edges $v \leftarrow u_i$ provide information that is not consistent with the current voting order.

In a RfA we are predicting a vote v given the previous voters U, the current session graph S and the follow graph F, as seen in Algorithm 5. We utilize the code provided by Tatti [52] to compute the agony of a unsigned weighted directed network as required by Algorithm 4.

The update rule for the follow graph F is similar to that for the agreement graph. We assume that every edge $e = (u, v) \in E_F$, has the attributes $e.follow$ and $e.common$, the follower ratio and the number of elections where u voted after v respectively. After a RfA voting session, the session graph S can be used to update the follower ratio and the corresponding weight as show in Algorithm 8. This allows us to bootstrap the model by beginning with an empty follow graph F. As the model predicts RfAs, the follow graph is updated and contains more information to predict the next RfA.

Algorithm 8: Update Follow graph after a session

Input: Session graph S, Candidate c, Follow graph F
Result: Updated Follow graph F

```
// Get all voters
```
$O \leftarrow V_S - \{c\}$
Order O by timestamp
for $v \in O$ **do**
  $vote_v \leftarrow w_S((v, c))$
  foreach *u who voted after v* **do**
    $vote_u \leftarrow w_S((u, c))$
    $e \leftarrow (u, v)$
    if $e \in E_F$ **then**
      $follow \leftarrow e.follow$
      $common_{uv} \leftarrow e.common$
      if $vote_v = vote_u$ **then**
        $follow \leftarrow ((follow \cdot common_{uv}) + 1)/(common_{uv} + 1)$
      else
        $follow \leftarrow (follow \cdot common_{uv})/(common_{uv} + 1)$
      end
      $common_{uv} \leftarrow common_{uv} + 1$
    else if $vote_u = vote_v$ **then**
      `// if` e `is a new edge`
      $common_{uv} \leftarrow$ number of elections where u voted after v
      $follow \leftarrow 1/common_{uv}$
      $E_F \leftarrow E_F \cup \{e\}$
    end
    $e.follow \leftarrow follow$
    $e.common \leftarrow common_{uv}$
    $w_F(e) \leftarrow e.follow - 0.5$
  end
end
return F

5.4 Voting Order Experiments

The iterative models based on balance and status theory predict votes in the order that they were cast. In this thesis, we wish to analyse the importance of the voting order for the quality of predictions from the iterative models. To achieve this, we gather two more RfAs that occurred in May and August of 2019. These RfAs are not part of the WIKI-RFA dataset, and therefore the models trained on the dataset will have not seen the votes in these session before.

The first is the RfA of user *HickoryOughtShirt?4*, that was completed on 1st May 2019. The RfA was successful with 182 support, 19 oppose and 9 neutral votes. This is an example of a RfA that did not have much opposition and consensus was evident in the proceedings. This RfA can be used to test if the model is able to effectively predict the minority of negative votes that appeared in this election, which were only 9% of all votes cast.

The second RfA we collected was the unsuccessful nomination of the user *Hawkeye7* in August 2019. In fact, this was the third RfA nomination for the user. He was successful in his first nomination in November 2009 and was promoted to an administrator. After that, he lost his administrative privileges following an ArbCom decision for misuse of his administrative tools. The second nomination in February 2016 resulted in failure even after receiving a significant amount of support votes (191 support and 95 opposition votes). The third nomination in August 2019 also resulted in failure after a fairly close voting phase. He received 91 support, 83 oppose and 15 neutral votes. This RfA is a perfect example of how Wikipedia RfAs are not a majority voting election. Therefore, it will be useful to study if the iterative models are able to effectively generalize the information that have learnt from the WIKI-RFA dataset. Also, we can analyse the impact of the order of votes to see if that can affect the prediction in especially close RfAs.

We first predict the votes in both RfAs in the same order that they took place in. We call this the *normal vote ordering*. Next, we reverse the order of votes from the second vote cast. This is because the first votes cast in Wikipedia RfAs are of the nominators and they provide the starting point for the iterative predictions. We refer to this ordering as *reversed vote ordering*. Lastly, we randomly permute the votes, except the first one cast by the co-nominator. We do 10 trails and then average the results. This is called *random voting order*.

By studying the predictive quality of both the status and balance based iterative models, we can understand the role of the voting order in each approach. We can also gain insights into creating a more global framework of

vote prediction if the voting order does not vastly affect the model's predictive accuracy.

5.5 Evaluation Metrics

In this section, we discuss the various metrics that we can use to evaluate the implementation of the models discussed in the previous sections. As mentioned in Section 5.1, the WIKI-RFA dataset has an imbalance of support votes. Therefore, simple measures such as *accuracy* scores might be misleading as the baseline accuracy for predicting all votes as support votes is nearly 73%. The models we implement in Section 5.2 and 5.2 output probabilities, and hence the metrics must also be able to utilize these outputs.

The independent vote prediction task is a binary classification task. The models implemented provide the probability of the vote being a support vote. Therefore, we have a target class $y \in \{-1, 1\}$ corresponding to oppose and support votes and the result is a probability $p \in [0, 1]$ for being a support vote. Hence, we propose traditional metrics such as Receiver Operator Characteristics (ROC) and Precision Recall (PR) to evaluate the results of the model. We also discuss how to compute F1 scores to evaluate model in a deterministic manner for a given threshold θ.

We can choose a threshold θ, for the probabilities that we have as the output from the model. Then, we predict all outputs where $p > \theta$ as $+1$ and where $p \leq \theta$ as -1. When we compare our predictions with the true outputs y, we get four possible outcomes. First, when the prediction is $+1$ and the true outcome is also $+1$, then it is called a *true positive* (TP). Second, when the prediction is -1 and the true output is also -1, then it is a *true negative* (TN). However, if the predicted output is -1 and the true output is $+1$, then it is referred to as a *false negative* (FN). Similarly, if the prediction is $+1$, but the true output is -1, then it is a *false positive* (FP). These four values can be represented in a *confusion matrix*, as seen in Figure 5.2.

5.5.1 Receiver Operating Characteristics

Now, the *true positive rate* (TPR) is the measure of the number of correct positive predictions made out all the available true positive outcomes and is defined as, $TPR = TP/(TP + FN)$. Similarly, the *false positive rate* (FPR) measures the number of incorrect classifications of negative samples out of all the available negative samples, i.e., $FPR = FP/(FP + TN)$. Therefore, the ROC curve is the space defined by the TPR as a function of the FPR, i.e., the TPR on y-axis and FPR on the x-axis. Each point on the ROC

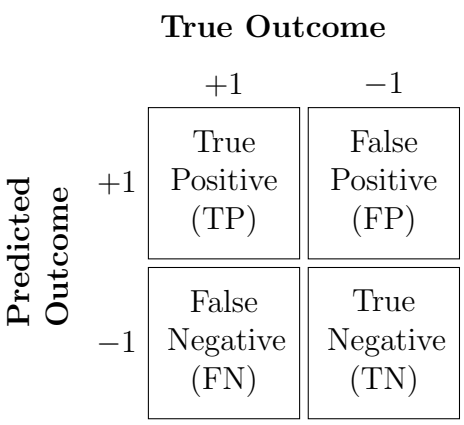


Figure 5.2: Confusion Matrix for binary classification task

curve corresponds to a confusion matrix at some threshold. A model that randomly predicts outcomes will plot a diagonal line, indicating that the TPR and FPR are equal. A perfect classifier's plot would have a point at $(0, 1)$, which indicates that there are no samples that are misclassified. Although the ROC curve can be visually inspected to compare models, we utilize the *area under the ROC curve* (AUC-ROC) as a quantitative measure of a model's performance. Therefore, the baseline random model has a AUC-ROC of 0.5.

The AUC-ROC score is unaffected by an imbalanced dataset. This means that a high AUC-ROC score might hide the fact that the baseline accuracy of predicting all samples as positive might indeed be higher than 0.5. Therefore, we need to be careful when interpreting the quality of the model solely based on the AUC-ROC score.

5.5.2 Precision Recall

Recall is the same as true positive rate (TPR), i.e., $recall = TP/(TP + FN)$. The ratio of the number of true positive predictions out of all the predicted positive outcomes is called *precision* (or positive predictive rate). It is defined as, $precision = TP/(TP + FP)$. Precision and recall are in tension, i.e., improving precision reduces recall and vice versa. Therefore, the Precision-Recall (PR) curve is the space defined by representing precision as a function of recall, i.e., precision on y-axis and recall on the x-axis. Each point on the PR curve corresponds to a single confusion matrix obtained from a particular value of the threshold θ. The baseline for the PR curve is based on the frequency of the positive label in the true outcomes and appears as a horizontal line in the plots. Hence, the PR curve is affected by the imbalance present in the dataset. Therefore, we define two measures to better represent

the imbalances present in the WIKI-RFA dataset.

The PR curve is usually defined with respect to the positive label probability. We refer to this curve as the positive PR curve and denote it by PR_{pos}. The positive baseline is computed as ratio of the true positive outputs and the total number of samples, $\text{baseline}_{\text{pos}} = (TP+FN)/(TP+FN+FP+TN)$. This will be higher for the WIKI-RFA dataset as there are more positive samples, i.e., support votes. We can measure the positive label performance by computing the area under the PR_{pos} curve (AUC-PR_{pos}), it is also called average precision score. The AUC-PR_{pos} score should be higher than the positive baseline to be significant. Even so, the AUC-PR_{pos} does not tell us if the model has learnt to predict negative votes equally well.

For this purpose, we define the negative PR curve as the PR curve where we consider the probability of predicting a negative outcome and denote it by PR_{neg}. As we have a binary classification task, if the positive probability vector is $\mathbf{p}$, then the negative probability vector is simply $\mathbf{1}-\mathbf{p}$. Now considering -1 to be the positive label, we can plot a PR_{neg} curve in the same manner. The negative baseline for the PR_{neg} curve is defined as the ratio of the true negative samples divided by the total number of samples, $\text{baseline}_{\text{neg}} = (TN+FP)/(TP+FN+FP+TN)$. As the negative samples, i.e., oppose votes, are the minority in the WIKI-RFA dataset, the corresponding negative baseline will also be lower. We can measure the performance of the model in predicting negative samples by computing the area under the PR_{neg} curve (AUC-PR_{neg}). This measure will be more important for evaluating the performance of the iterative models on the WIKI-RFA dataset.

5.5.3 F1 Score

The *F1 score* is the harmonic mean of precision and recall and defined as

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Therefore, if we consider a PR curve, then the F1 score is computed by taking the precision and recall values at a particular point on that curve. If the curve was for the positive class probability, i.e., a PR_{pos} curve, then we define the associated F1 score as the F1_{pos} score. Similarly, if the curve is for the negative class probabilities, i.e., a PR_{neg} curve, then the score is called the F1_{neg} score. A simple average of the F1_{pos} and F1_{neg} scores is called the *macro F1 score*, and is defined as follows,

$$\text{F1}_{\text{macro}} = \frac{\text{F1}_{\text{pos}} + \text{F1}_{\text{neg}}}{2}.$$

The $F1_{macro}$ score is useful for datasets which are imbalanced as it places equal weight on the performance for both positive and negative labels.

All the metrics in the previous subsubsection used probabilities directly to evaluate the overall performance of the model. As the F1 score is calculated for a point in the PR curve, it corresponds to the particular value of threshold θ, that yielded those values of precision and recall in the confusion matrix. Therefore, we can now plot the F1 score as a function of the threshold. This allows us to analyse both the $F1_{pos}$ and $F1_{neg}$ plots versus the threshold and choose the optimal value of θ, that maximizes the $F1_{macro}$ score. Hence, we can understand how the model will perform when asked to deterministically predict classes.

Chapter 6

Results and Discussion

In this chapter, we present the results of the experiments described in Chapter 5 and discuss the performance of the models. First, in Section 6.1, we describe the development, training and testing split of the dataset. Then, we present the results of the graph combination model and discuss the most important features of the logistic regression classifier. Moreover, we compare it to the performance of the iterative models in the same test dataset and explain the shortcoming of the graph combination model. Second, we display the results of the iterative models using the entire WIKI-RFA dataset in Section 6.2. Further, we examine the performance of the iterative models and discuss the optimal selection of the threshold to predict results. Lastly, the results of the voting order experiments are presented in Section 6.3. We analyse the significance of the voting order on the performance of the iterative models.

6.1 Test Dataset Results

As we described in Section 5.2.2, the graph combination model requires the WIKI-RFA dataset to be split into three part to prevent data leak. Although we performed the experiments for many variation of these three splits, we will show the results from the $30 - 30 - 40$ split into development (dev), training (train) and testing (test) respectively. As the model aims to predict votes, we choose to split it on the percentage of votes, as seen in Table 6.1. We round up the nearest RfA ending so that we have contiguous elections in each split. In Table 6.1, we see this as small overlaps between the last dates of the previous splits and the first dates. The time frame overlap is almost exactly seven days, which is the duration of a RfA. Next, we present the details of the auxiliary and signed graphs formed from the dev dataset

and the graph combination model's performance on the test dataset.

Table 6.1: WIKI-RFA dataset split information

Feature	Development	Training	Testing
Percentage	30%	30%	40%
Number of votes	62833	62807	83830
Number of RfAs	1668	1551	1314
First Date	22/02/2004	31/10/2006	24/06/2008
Last Date	06/11/2006	30/06/2008	01/01/2019

Next, we also show the results of the iterative models on the test dataset. We achieve this by evaluating the iterative models' results in the same time period as the test data. Through this approach, we can compare the benefits of the iterative model, which can utilize both the development and training datasets to learn and update its respective relationship graph.

We provide the evaluation metrics for all models along with the baseline for the test dataset, as seen in Table 6.2. The AUC-PR_{neg} baseline shows that negative votes are the minority in the test test. Similarly, the AUC-PR_{pos} baseline shows that a model predicting all votes as support votes can achieve nearly 77% accuracy. Now, we discuss the results of each model in more detail.

Table 6.2: Results of different models for the test split of the WIKI-RFA dataset

Model	AUC-ROC	AUC-PR_{pos}	AUC-PR_{neg}
Baseline	0.5	0.776	0.224
Graph Combination	0.542	0.798	0.251
Iterative Balance	0.815	0.922	0.614
Iterative Status	0.754	0.9	0.486

6.1.1 Graph Combination Model Results

We start by describing the details of the auxiliary and signed graphs formed from the dev dataset, as seen in Table 6.3. The *% of test users covered* refers the percentage of unique users in the test dataset present in the graph. It can be used as a proxy to measure the amount of information a graph can provide for predicting a vote in a RfA in the test dataset. We see that the

similarity graph is fairly dense and is completely connected, as we chose the minimum similarity of an edge of 0.03 to be considered viable. It also has the largest coverage of nodes in the test dataset. The *talk graph* also has a large strongly connected component (as it is directed) and a smaller test user coverage. The *social interaction graph* is the same as the talk graph, but is unweighted, therefore, has the same statistics as the talk graph. As we explained in Section 5.2.1.2, we also include the reversed talk and social interaction graphs to gain additional features. The *signed graph* is by far the smallest, least dense and weakest connected graph of all the graphs. This is because, the signed graph only contains the voting data from the dev dataset.

Table 6.3: Information of graphs formed using development data split

Graph	$\|V\|$	$\|E\|$	density	largest component size	% of test users covered
Topic Similarity	6368	1463465	0.0721	6368	27.3
Talk	5477	213307	0.0071	3489	18.9
Signed Voting	4675	65595	0.003	1083	9

Using these auxiliary and signed graphs we prepare the training and testing feature matrices $\mathbf{X}$ and $\mathbf{X}_{text}$ and target vectors, $\mathbf{y}$ and $\mathbf{y}_{test}$ respectively. We train the *Logistic Regression* (LR) model on the training feature matrix and target vector using five fold cross validation. The feature importances of the trained LR model are shown in Figure 6.1. We see both, the five auxiliary features and the 36 triadic features. *Talk Graph R* and *Social Interactions R* features refer to the reversed versions of the talk and social interaction graphs respectively. We see that the topic similarity graph has the largest coefficient. The importance of the similarity feature amongst the auxiliary features can be explained by the fact that, the topic similarity has the largest coverage of test users and therefore contributes the most information.

Among the other features, the triad *FB++* has the next largest coefficient. This result is consistent with balance theory, which would predict a positive edge to maintain the balance in the triad. However, for this triad, status theory does not have a preference of either a positive or negative edge. Attempting to interpret the result in terms of status theory we have: if a candidate and voter have a mutual friend who they both respect, then it is more likely that the voter will support the candidate. Though this is not typically expected behaviour, it might suggest some subtle social influences at play amongst the voters.

The other triadic feature that is significant is the triad *FB−+*. Yet, here

the coefficient is negative, indicating that the prediction is more likely to be a negative edge, i.e., an oppose vote. This result is again consistent only with balance theory. Balance theory predicts the vote is negative to balance the resulting triad to be balanced. Status theory implies that, if the candidate has a common friend, whom the voter does not respect, but the friend looks up to the candidate, then the voter is undecided. However, the result in this case indicates that the voter has a negative view of the candidate and votes against them more often. Therefore, we see the results agreeing more strongly with balance theory rather than status theory.

And then, we see that both versions of the social interaction graphs are more significant than the talk graph. This indicates that simple existence correspondence is more important than the amount of correspondence or the direction of correspondence between voters and their voting neighbourhood.

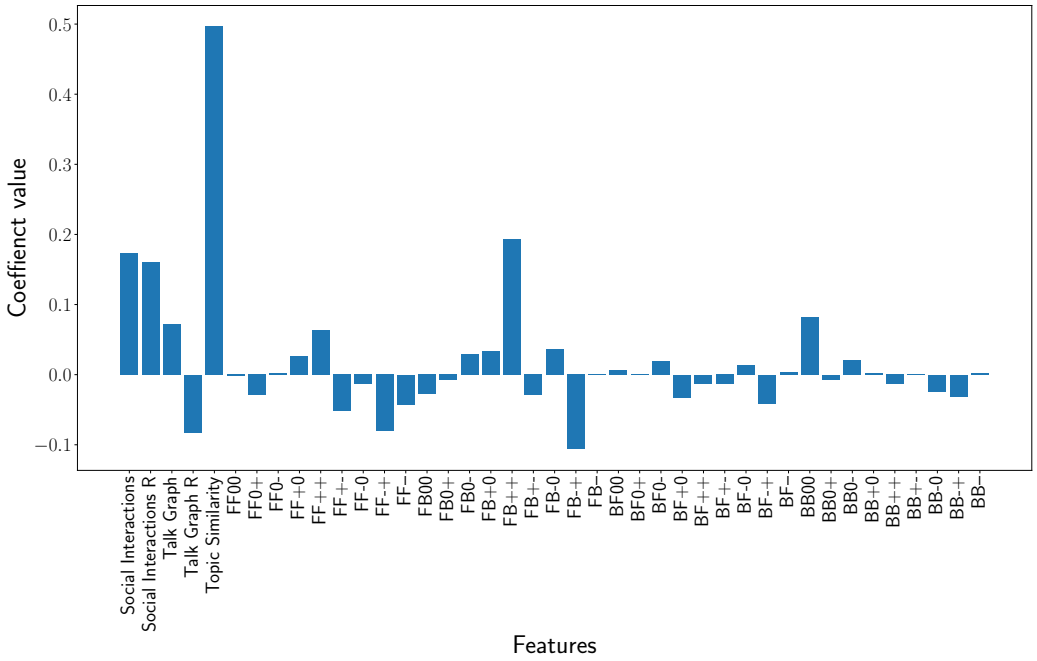


Figure 6.1: Feature importances of the trained Logistic Regression model

Then, we tested the trained LR model on the test feature matrix and evaluated the output with the target vectors. The results and the evaluation metrics are seen in Table 6.2. We see that the graph combination model does not perform very well. It has marginal gains on all three baseline metrics. We can analyse these in more details looking at the ROC and PR curves, shown in Figure 6.2. The ROC-AUC curve shows that model has a marginal improvement over the 0.5 baseline. Similarly, the PR_{neg} curve depicts that

the model fails to learn to predict negative votes more than the baseline of 0.2. This combined with the marginal gain in predicting positive votes implies that the model has not learnt any statistically significant information.

We can explain the low performance of the model by using its lack of information. As the features that are created for each vote are dependent on only the dev dataset, they have limited impact when predicting votes in the test dataset. This is due to the fact that RfAs are chronological, as there are many more newer users in the test dataset and there is no information available on them in the dev dataset. This problem might reduce, if we change the percentage of data in the dev, train and test datasets. However, increasing the size of dev dataset split, leads to a lack of training data. In this scenario, the LR model cannot efficiently learn the coefficients from the features that have now possess more information. This leads again to the model only achieving marginal improvements over the baselines.

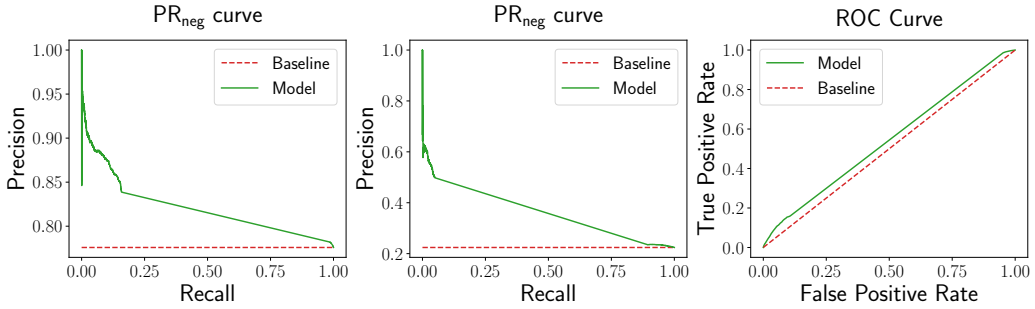


Figure 6.2: Logistic Regression plots for test data

6.1.2 Iterative Model Results

Now, we discuss the results of the iterative models for the test dataset. We see the evaluation metrics for both the iterative balance and status models in Table 6.2. These results are the predictions the iterative models makes for the RfAs present in the test dataset. We see that both models perform much better than the baselines as well as the graphical combination model.

However, directly comparing the iterative models' result to the graphical model's result is not fair. Because, the iterative models assimilate the information each election as they progress in the test dataset and utilize that information to predict votes in the next RfA. Therefore, the predictions of the iterative model are not independent. In spite of this, analysing the improvements of the iterative models provides understanding of the inherent shortcomings of the graphical combination model discussed previously.

Out of both models, we see that the *Iterative Balance Model* performs much better. We see that the model is able to better predict both positive and negative votes, seen by the large AUC-PR_{pos} and AUC-PR_{neg} scores respectively. This is in line with the previous analysis that balance theory better predicts triads in the voting neighbourhood. Here, we see that the LSN of the voter conforms more according to balance theory than status theory.

Another analysis, both the iterative balance and status models achieve better performance than the graphical combination model utilizing only the voting data. Furthermore, this indicates that there is a scope of incorporating the auxiliary features to the iterative models to further improve the performance of the models. Moreover, it clearly shows that solving the lack of information problem present in the graph combination model can lead to better predictions.

We can analyse the iterative models further considering the complete WIKI-RFA dataset.

6.2 Complete WIKI-RFA Results

The iterative models described in Second 5.3, can be bootstrapped to predict all the votes in the WIKI-RFA dataset. The results for the models along with the baselines are shown in Table 6.4. We see the complete dataset is more imbalanced than the test dataset, seen by the larger AUC-PR_{pos} and smaller AUC-PR_{neg} baselines.

6.2.1 New Voter Analysis

As discussed in Section 5.3, we marked all votes that were predicted without any information when we encountered new voters. This amounted to 11812 or approximately 5.7% of all votes predicted. The distribution of the true value of these votes is 9217 support and 2595 oppose votes. This shows that new voters are almost 3x more likely to vote positively. We analysed these new voters' votes with the progress of the election of the time to study herd mentality. Comparing these votes to the sign of the cumulative sum of votes until that point, we see that nearly 81% of new votes are the same as the herd. Similarity, we also compare the new voters to the votes cast by the person immediately before them. We see that 76% of new voters agree with the previous voter. Therefore, we can adopt a simple strategy of predicting the new voters to have a probability of voting support equal to the fraction of support votes cast until that point.

Table 6.4: Results of iterative models on the complete WIKI-RFA dataset

Model	AUC-ROC	AUC-PR_{pos}	AUC-PR_{neg}
Baseline	0.5	0.784	0.216
Iterative Balance	0.835	0.935	0.635
Iterative Status	0.784	0.917	0.502

Table 6.5: Information of relationship graphs of iterative models using entire WIKI-RFA dataset

Relationship Graph	$\|V\|$	$\|E\|$	density	largest component size
Agreement Graph	11924	2451028	0.0345	11908
Follow Graph	11924	3136303	0.0220	11563

6.2.1.1 Iterative Balance Model Results

The iterative balance models performs very well even when predicting all the votes in the entire WIKI-RFA dataset. The results in Table 6.4, shows that on every metric the balance model has a significant margin over the baseline. Especially, we see that negative votes are predicted almost three times better than the baseline, seen by the AUC-PR_{neg} score of 0.635. This indicates that the model has collected useful information in the *agreement graph*, as seen in Table 6.5. The graph obtained at the end of the process is fairly large and dense and is nearly connected. Therefore, a rich representation of relationships between Wikipedia users is stored in the agreement graph.

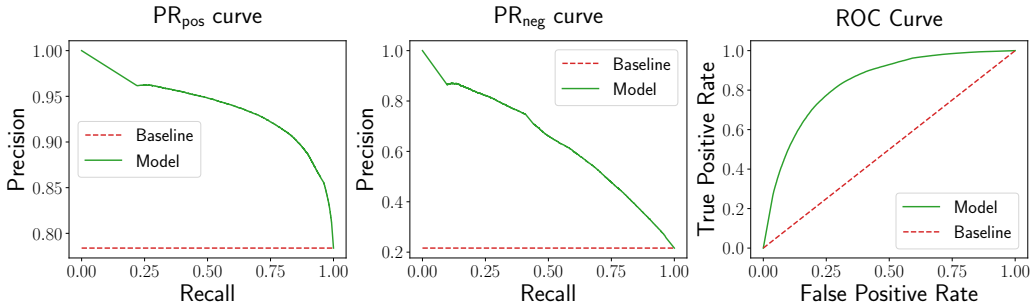


Figure 6.3: Plots for the Iterative Balance Model on the complete WIKI-RFA dataset

The plots in Figure 6.3, show that the model consistently performs well above the baselines. Now, in choosing an optimal threshold for the model,

we turn to the plots in Figure 6.4. These show how the F1 score changes as we move the threshold parameter θ. We see that the $F1_{pos}$ score only starts to decrease gradually after $\theta = 0.5$. Also, there is a peak for the $F1_{neg}$ score a little after the point of $\theta = 0.5$. Therefore, we can choose $\theta = 0.53$, to obtain a $F1_{pos} = 0.887$ and $F1_{neg}$ =0.602. This gives us $F1_{macro} = (0.887 + 0.602)/2 = 0.745$. This threshold also indicates that even if λ_1^+ of the LSN is slightly smaller than λ_1^- then the vote predicted is positive. Therefore, the model has good compliance with balance theory.

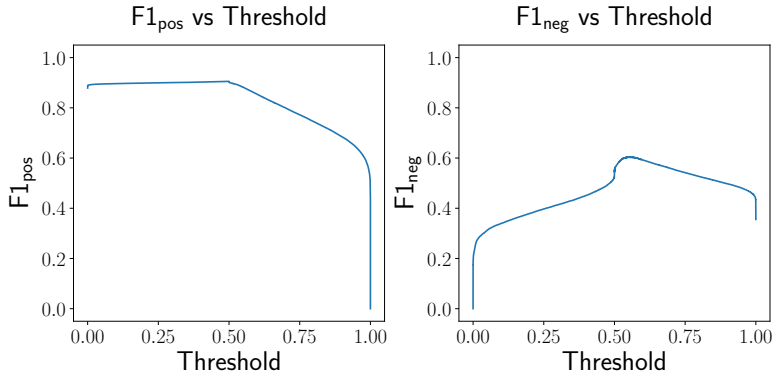


Figure 6.4: F1 score versus threshold plots for Iterative Balance Model

6.2.2 Iterative Status Model Results

Table 6.4 show that the iterative status model also performs admirably above the baseline results. We also see in Table 6.5, that the *follow graph* is large, fairly dense and has large strongly connected components. However, its performance is still relatively lower than the iterative balance model.

In Figure 6.5, we see that the PR_{pos} curve is nearly identical to that of the iterative balance model. This is also reflected in the high AUC-PR_{pos} score comparable to the iterative balance model. However, the PR_{neg} curve clearly shows that there is a lower quality when predicting negative votes. This translates in the smaller AUC-PR_{neg} score and explains the overall lower AUC-ROC score of the model. The lower predictive performance can be explained using our earlier analysis of the graph combination model. We see that in reality, the triads where status theory is ambivalent actually have a preference for a particular sign. Therefore, the cases when the agony of the LSN is equal for both cases, i.e., $\alpha^+ = \alpha^-$, should not map to $p = 0.5$. Rather, we must modify status theory to better represent signed relationships in a network.

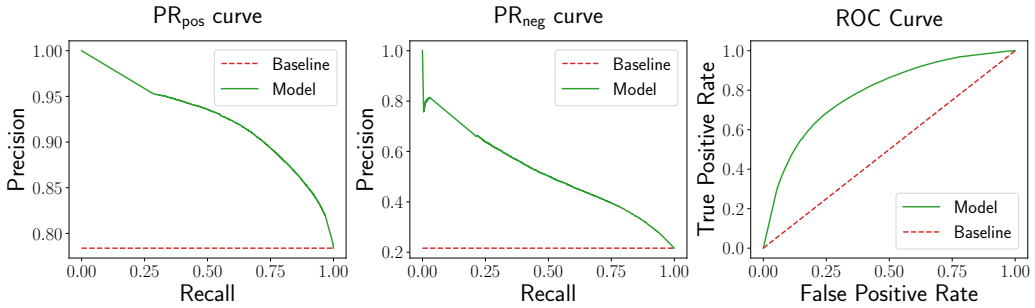


Figure 6.5: Plots for the Iterative Status Model on the complete WIKI-RFA dataset

In Figure 6.6, we see another issue caused by the large number of the support vote probabilities being 0.5. The $F1_{pos}$ curve is again identical to the $F1_{pos}$ curve of the iterative balance model. However, the $F1_{neg}$ curve shows that there is an inflection at $\theta = 0.5$. The change is quite drastic with $F1_{neg} = 0.05$ at $\theta = 0.49$ and $F1_{neg} = 0.32$ at $\theta = 0.5$. This affects the choice of an optimal threshold for the model. We see that the $F1_{neg}$ score increases as threshold is increased beyond $\theta = 0.5$. This indicates that closer to 0.5, there are many false positives. However, choosing $\theta = 0.75$ at the peak of the $F1_{neg}$ score is not suitable, as the $F1_{pos}$ score starts to drop much more significantly. Hence, we choose $\theta = 0.63$ as the constrained optimum giving us $F1_{pos}$ =0.861 and $F1_{neg}$ =0.504, therefore, $F1_{macro} = 0.606$. This deterministic metric is also lower than the 0.745 of the iterative balance model. The choice of threshold close ot $= 0.6$ suggests that the α^+, the agony for the positive vote case, must be considerably lower than α^-, the agony for the negative vote case, to predict a positive vote. Therefore, this also indicates that we need to make additional modifications to status theory if we want to increase the predictive power of the iterative status model.

6.3 Voting Order Results

In Section 5.4, we discussed how to study the effects of voting order on the performance of the iterative models trained on the complete WIKI-RFA dataset. The results for the third RfA nomination of user *Hawkeye7* is presented in Table 6.6 and is referred to the *failed RfA*. Similarly, the results of user *HickoryOughtShirt?4*'s nomination is show in table 6.7 and is called the *successful RfA*.

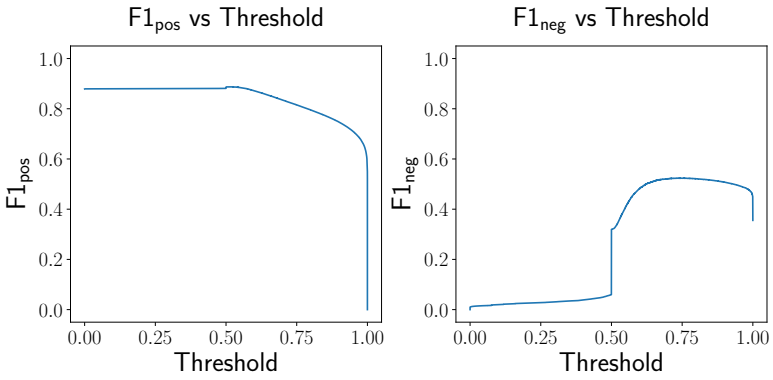


Figure 6.6: F1 score versus threshold plots for Iterative Status Model

6.3.1 Failed RfA Results

We see that the voting in the failed RfA is fairly tight. The AUC-PR_{pos} score is 0.52 and AUC-PR_{neg} is 0.479, indicating that the election did indeed have more support votes than oppose. Nevertheless, the RfA failed as the Bureaucrat responsible concluded that the consensus was to not promote the user.

For the *normal voting order*, as seen in Table 6.6, the balance model performs much worse when compared to the status model. In fact, the iterative balance model performs below the baseline for all metrics. This can be attributed to the fact that this particular user had two previous nominations of which one was successful. Therefore, a symmetric measure of agreement might not be helpful in predicting the balance as the election progresses, especially when the voting margins are tight. On the other hand, we see that the iterative status model performs better the iterative balance model, but is still worse than the baseline metrics for a random model. This clearly shows that both models are struggling to predict votes in a RfAs with a narrow margin of difference between support and oppose votes.

For the *reversed voting order*, we see that iterative balance model performs considerably better compared to the normal voting order. Especially, we see that all the metrics, i.e., AUC-ROC, AUC-PR_{pos} and AUC-PR_{neg} scores have improved above the baseline, increasing the model's overall performance. This is interesting, as it suggests that the LSNs formed by reversing the voting order provides better information on the voting behaviour than the actual voting order. Similarly, we see the status model also gaining in performance when the voting order is reversed. Although the AUC-PR_{neg} score is still below the baseline, we see the overall performance has improved. We can explain the model's difficulty in predicting negative votes using our previous

analysis; status theory complies less with the true data than balance theory. Therefore, we infer that reversing the voting order improves the performance of both models.

Meanwhile, the results of the average of ten *random voting orders* for both models lie in between the results for the normal and reversed voting methods. We see that the result for the iterative balance model is above the baseline for all metrics and the iterative status model is close to the baseline. Therefore, we see that the voting order does affect the performance of the iterative models and that we can benefit from reversing the voting order and averaging the results to obtain better predictions for RfAs that have tight margins.

Table 6.6: Results for different vote orderings for the failed RfA

Model	Vote Order	AUC-ROC	AUC-PR_{pos}	AUC-PR_{neg}
Baseline	-	0.5	0.52	0.479
Iterative Balance	Normal	0.392	0.490	0.403
	Reversed	0.575	0.606	0.529
	Random	0.527	0.552	0.517
Iterative Status	Normal	0.454	0.532	0.457
	Reversed	0.515	0.563	0.466
	Random	0.493	0.538	0.480

6.3.2 Successful RfA Results

In Table 6.7, we see that the result of the RfA is clearly evident in the AUC-PR_{pos} and AUC-PR_{neg} baselines. Nearly, all votes are supporting the candidate and there are a few minority opposition votes. For this RfA, we see the results are in line with the results we obtained for the failed RfA.

For the *normal voting order*, we see that the iterative balance model has AUC-ROC and AUC-PR_{pos} scores lower than the baseline but AUC-PR_{neg} scores well above the baseline. This indicates that the model is better able to predict oppose votes, but at the cost of the better predictions for the support votes. We see a similar phenomenon for the iterative status model where the AUC-PR_{pos} is below the baseline but the overall AUC-ROC is slightly above the random model baseline. This highlights the difficulty of predicting the oppose votes in a fairly clear election.

Once again, in the results for the *reversed voting order*, we see that both iterative models have better performance across all metrics. Especially, we

see both models have much larger AUC-PR_{pos} scores which in turn boosts the AUC-ROC scores, as the positive votes are the majority in this RfA. As a result, we also see that the AUC-PR_{neg} scores drop, indicating the tension in optimizing both positive and negative vote prediction.

The average of 10 *random voting order* results show that the performance of the models are overall better than the normal voting order.

A note to consider is that this analysis and results are only for the two new RfAs and can be further analysed in detail as a separate project.

Table 6.7: Results for different vote orderings for the successful RfA

Model	Vote Order	AUC-ROC	AUC-PR_{pos}	AUC-PR_{neg}
Baseline	-	0.5	0.905	0.095
Iterative Balance	Normal	0.48	0.90	0.231
	Reversed	0.649	0.933	0.216
	Random	0.607	0.921	0.273
Iterative Status	Normal	0.503	0.898	0.211
	Reversed	0.628	0.931	0.151
	Random	0.612	0.921	0.230

Chapter 7

Conclusions and Future Work

In this thesis, we described how voting in a community can be intuitively and structurally represented using signed networks. The positive and negative links directly map to a voter's support for or opposition to a candidate respectively. Furthermore, we discussed how the problem of vote prediction is related to the task of predicting the sign of a link in the signed network. Then, we explain the structural theories of balance and status in signed networks, derived from the field of social psychology. We discuss the existing approaches and methods to solve the sign prediction tasks and explain their limitations for the vote prediction task. Therefore, we proposed two new models for the task of vote prediction, incorporating the chronological nature of voting.

The first, is an extension of supervised models for sign prediction approaches. To predict the vote of an individual voter towards a candidate, we introduce the concept of a *voting neighbourhood*. We gather features from the voting neighbourhood of a voter and various auxiliary graphs that represent relationships between the members in community. Then, we count the unique triadic features that capture the theories of balance and status in the voting neighbourhood. The combination of all these features and the known true votes form a traditional supervised machine learning problem. Therefore, any linear classification model can be trained to predict positive and negative edges that correspond to support or oppose votes.

The second, we proposed an iterative model that relies solely on structural balance and status theories. We extend upon the concept of the voting neighbourhood to create a *Local Signed Network* (LSN) for a voter. We state that within this LSN, the voter prefers to vote in a manner that, the resultant LSN better adheres to either balance or status theory. Therefore, we arrive at two models, the first in which a voter maintains the balance of the LSN and the second where the voter preserves the status in the LSN. Then, we

developed methods to quantitatively measure a network's adherence with either balance or status theory. We utilize these measures to predict how an independent user will vote given their LSN. Furthermore, as the voting progresses, we proposed to maintain a relationships graph that encapsulates the interaction between the voters in the community. This, led to the ability to iteratively predict votes in a session and assimilating the information at the end of a session.

From the results presented in Chapter 6, we conclude that the proposed models are effective in predicting the votes in Wikipedia RfAs. The graph combination models perform poorly only due the the restrictions of the supervised learning framework and lack of information to predict votes far off in the future. We show that the iterative models overcome these problems and are able to effectively predict votes of the entire dataset using only voting related data. We analysed how new voters with respect to the model, tend to vote. This showed that a majority new users follow the herd mentality or vote similar to person immediately before them. Furthermore, we show that votes in the promotion process of Wikipedia administrators are generally more compliant with balance theory than status theory. We also conclude that the iterative models are sensitive to the voting order. Reversing the order of voting brings greater predictive power in both elections that are close as well as elections that are landslides.

Future work includes developing a modified theory of social status in signed network that better represents relationships between people in real life. Similarly, we plan to incorporate external non-voting features to improve the iterative models and extend the experiments to congressional and parliamentary voting on bills and laws.

Bibliography

[1] Agrawal, P., Garg, V. K., and Narayanam, R. Link label prediction in signed social networks. In *Twenty-Third International Joint Conference on Artificial Intelligence* (2013).

[2] Ahmadalinezhad, M., and Makrehchi, M. Sign prediction in signed social networks using inverse squared metric. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation* (2018), pp. 220–227.

[3] Ali, S. N., and Kartik, N. A theory of momentum in sequential voting.

[4] Arinik, N., Figueiredo, R., and Labatut, V. Signed graph analysis for the interpretation of voting behavior, 2017.

[5] Asim, Y., Niazi, M. A., Raza, B., and Malik, A. K. Personal vs. know-how contacts: which matter more in wiki elections? *arXiv preprint arXiv:1804.07450 6*, 1 (2018), 4.

[6] Banerjee, A. V. A simple model of herd behavior. *The quarterly journal of economics 107*, 3 (1992), 797–817.

[7] Brito, A. C. M., Silva, F. N., and Amancio, D. R. A complex network approach to political analysis: Application to the brazilian chamber of deputies. *PLOS ONE 15*, 3 (2020).

[8] Budhwar, A., Kuboi, T., Dekhtyar, A., and Khosmood, F. predicting the vote using legislative speech. In *Proceedings of the 19th Annual International Conference on Digital Government Research* (2018), p. 35.

[9] Burke, M., and Kraut, R. Mopping up: modeling wikipedia promotion decisions. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (2008), pp. 27–36.

[10] Cabunducan, G., Castillo, R., and Lee, J. B. Voting behavior analysis in the election of wikipedia admins. In *2011 International Conference on Advances in Social Networks Analysis and Mining* (2011), pp. 545–547.

[11] Cartwright, D., and Harary, F. Structural balance: a generalization of heider's theory. *Psychological Review 63*, 5 (1956), 277–293.

[12] Cesa-Bianchi, N., Gentile, C., Vitale, F., and Zappella, G. A correlation clustering approach to link classification in signed networks. In *Annual Conference on Learning Theory* (2012), vol. 23, Microtome, pp. 34–1.

[13] Chiang, K.-Y., Hsieh, C.-J., Natarajan, N., Dhillon, I. S., and Tewari, A. Prediction and clustering in signed networks: a local to global perspective. *Journal of Machine Learning Research 15*, 1 (2014), 1177–1213.

[14] Chiang, K.-Y., Natarajan, N., Tewari, A., and Dhillon, I. S. Exploiting longer cycles for link prediction in signed networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), pp. 1157–1162.

[15] Davis, J. A. Structural balance, mechanical solidarity, and interpersonal relations. *American Journal of Sociology 68*, 4 (1963), 444–462.

[16] Derr, T., and Tang, J. Congressional vote analysis using signed networks. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (2018), IEEE, pp. 1501–1502.

[17] Derthick, K., Tsao, P., Kriplean, T., Borning, A., Zachry, M., and McDonald, D. W. Collaborative sensemaking during admin permission granting in wikipedia. In *OCSC'11 Proceedings of the 4th international conference on Online communities and social computing* (2011), pp. 100–109.

[18] Diestel, R. *Graph Theory*. 1997.

[19] Gallier, J. Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey. *arXiv preprint arXiv:1601.04692* (2016).

[20] Gerrish, S., and Blei, D. M. Predicting legislative roll calls from text. In *Proceedings of the 28th International Conference on Machine Learning* (2011), pp. 489–496.

[21] Gu, S., Chen, L., Li, B., Liu, W., and Chen, B. Link prediction on signed social networks based on latent space mapping. *Applied Intelligence 49*, 2 (2019), 703–722.

[22] Guha, R., Kumar, R., Raghavan, P., and Tomkins, A. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web* (2004), pp. 403–412.

[23] Gupte, M., Shankar, P., Li, J., Muthukrishnan, S., and Iftode, L. Finding hierarchy in directed online social networks. In *Proceedings of the 20th international conference on World wide web* (2011), pp. 557–566.

[24] Harary, F. On the notion of balance of a signed graph. *Michigan Mathematical Journal 2*, 2 (1953), 143–146.

[25] Heider, F. Attitudes and cognitive organization. *The Journal of Psychology 21*, 1 (1946), 107–112.

[26] Hou, Y. P. Bounds for the least laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica 21*, 4 (2005), 955–960.

[27] Hsieh, C.-J., Chiang, K.-Y., and Dhillon, I. S. Low rank modeling of signed networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), pp. 507–515.

[28] Jankowski-Lorek, M., Ostrowski, L., Turek, P., and Wierzbicki, A. Modeling wikipedia admin elections using multidimensional behavioral social networks. *Social Network Analysis and Mining 3*, 4 (2013), 787–801.

[29] Karimi, H., Derr, T., Brookhouse, A., and Tang, J. Multifactor congressional vote prediction. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2019), pp. 266–273.

[30] Kazienko, P., Musial, K., Kukla, E., Kajdanowicz, T., and Bródka, P. Multidimensional social network: model and analysis. In *International Conference on Computational Collective Intelligence* (2011), Springer, pp. 378–387.

[31] Kearns, M. J., Judd, J. S., Tan, J., and Wortman, J. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences of the United States of America 106*, 5 (2009), 1347–1352.

[32] Khodadadi, A., and Jalili, M. Sign prediction in social networks based on tendency rate of equivalent micro-structures. *Neurocomputing 257* (2017), 175–184.

[33] Kordzadeh, N., and Kreider, C. Revisiting request for adminship (rfa) within wikipedia: How do user contributions instill community trust? *The Journal of the Southern Association for Information Systems 4*, 1 (2016), 1.

[34] Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., Luca, E. W. D., and Albayrak, S. Spectral analysis of signed graphs for clustering, prediction and visualization. In *SDM* (2010), pp. 559–570.

[35] Lee, J. B., Cabunducan, G., Cabarle, F. G., Castillo, R., and Malinao, J. A. Uncovering the social dynamics of online elections. *Journal of Universal Computer Science 18* (2012), 487–505.

[36] Leskovec, J., Huttenlocher, D., and Kleinberg, J. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web* (2010), pp. 641–650.

[37] Leskovec, J., Huttenlocher, D., and Kleinberg, J. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), pp. 1361–1370.

[38] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[39] Levorato, M., and Frota, Y. Brazilian congress structural balance analysis. *Journal of Interdisciplinary Methodologies and Issues in Sciences* (2016).

[40] Li, H. S., and Li, H. H. A note on the least (normalized) laplacian eighva; ue of signed graphs. *Tamkang Journal of Mathematics 47*, 3 (2016), 271–278.

[41] Liben-Nowell, D., and Kleinberg, J. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology 58*, 7 (2007), 1019–1031.

[42] Liu, S., Xiao, J., and Xu, X. Link prediction in signed social networks: from status theory to motif families. *IEEE Transactions on Network Science and Engineering* (2019), 1–1.

[43] MediaWiki. Api:usercontribs — mediawiki, the free wiki engine, 2019. [Online; accessed 22-March-2020].

[44] Meir, R., Gal, K., and Tal, M. Strategic voting in the lab: compromise and leader bias behavior. *Autonomous Agents and Multi-Agent Systems 34*, 1 (2020), 31.

[45] Picot-Clémente, R., Bothorel, C., and Jullien, N. Contribution, social networking, and the request for adminship process in wikipedia. In *Proceedings of the 11th International Symposium on Open Collaboration* (New York, NY, USA, 2015), OpenSym '15, Association for Computing Machinery.

[46] Picot-Clemente, R., Bothorel, C., and Jullien, N. Social interactions vs revisions, what is important for promotion in wikipedia? In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015* (2015), pp. 888–893.

[47] Putzke, J., and Takeda, H. Stated neutrality in voting networks–the case of wikipedia's request for adminship. In *ICIS* (2017).

[48] Shahriari, M., and Jalili, M. Ranking nodes in signed social networks. *Social Network Analysis and Mining 4* (12 2014).

[49] Tal, M., Meir, R., and Gal, Y. K. A study of human behavior in online voting. *adaptive agents and multi-agents systems* (2015), 665–673.

[50] Tang, J., Chang, S., Aggarwal, C., and Liu, H. Negative link prediction in social media. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2015), WSDM '15, Association for Computing Machinery, pp. 87–96.

[51] Tatti, N. Faster way to agony discovering hierarchies in directed graphs. In *ECMLPKDD'14 Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III* (2014), pp. 163–178.

[52] Tatti, N. Tiers for peers: a practical algorithm for discovering hierarchy in weighted networks. *Data Mining and Knowledge Discovery 31*, 3 (2017), 702–738.

[53] Wikipedia contributors. Wikipedia:notnow — Wikipedia, the free encyclopedia, 2018. [Online; accessed 5-May-2020].

[54] Wikipedia contributors. Wikipedia:arbitration committee — Wikipedia, the free encyclopedia, 2019. [Online; accessed 4-May-2020].

[55] Wikipedia contributors. Help:user contributions — Wikipedia, the free encyclopedia, 2020. [Online; accessed 4-May-2020].

[56] Wikipedia contributors. Wikipedia:administrators — Wikipedia, the free encyclopedia, 2020. [Online; accessed 4-May-2020].

[57] Wikipedia contributors. Wikipedia:bureaucrats — Wikipedia, the free encyclopedia, 2020. [Online; accessed 4-May-2020].

[58] Wikipedia contributors. Wikipedia:edit warring — Wikipedia, the free encyclopedia, 2020. [Online; accessed 2-May-2020].

[59] Wikipedia contributors. Wikipedia:ips are human too — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Wikipedia:IPs_are_human_too&oldid=940676725`, 2020. [Online; accessed 4-May-2020].

[60] Wikipedia contributors. Wikipedia:namespace — Wikipedia, the free encyclopedia, 2020. [Online; accessed 4-May-2020].

[61] Wikipedia contributors. Wikipedia:no original research — Wikipedia, the free encyclopedia, 2020. [Online; accessed 2-May-2020].

[62] Wikipedia contributors. Wikipedia:snowball clause — Wikipedia, the free encyclopedia, 2020. [Online; accessed 5-May-2020].

[63] Wikipedia contributors. Wikipedia:what wikipedia is not — Wikipedia, the free encyclopedia, 2020. [Online; accessed 2-May-2020].

[64] Wikipedia contributors. Wikipedia:wikipedians — Wikipedia, the free encyclopedia, 2020. [Online; accessed 4-May-2020].

[65] Yang, S.-H., Smola, A. J., Long, B., Zha, H., and Chang, Y. Friend or frenemy? predicting signed ties in social networks. In *Proceedings of the 35th International ACM SIGIR Conference on Research*

and Development in Information Retrieval (New York, NY, USA, 2012), SIGIR '12, Association for Computing Machinery, pp. 555–564.

[66] Yano, T., Smith, N. A., and Wilkerson, J. D. Textual predictors of bill survival in congressional committees. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2012), pp. 793–802.

[67] Yogatama, D., Heilman, M., O'Connor, B., Dyer, C., Routledge, B. R., and Smith, N. A. Predicting a scientific community's response to an article. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (Edinburgh, Scotland, UK., July 2011), Association for Computational Linguistics, pp. 594–604.

[68] Zaslavsky, T. Signed graphs. *Discrete Applied Mathematics 4*, 1 (1982), 47–74.

[69] Zou, J., Meir, R., and Parkes, D. Strategic voting behavior in doodle polls. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (2015), pp. 464–472.